VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Interaktivní průvodce webovými aplikacemi
Interactive Tour for Web Applications

2017

Bc. Jan Stočes

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jan Stočes**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Interaktivní průvodce webovými aplikacemi
Interactive Tour for Web Applications

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vyvinout samostatný systém umožňující realizaci interaktivních průvodců v rámci obecných webových aplikací. Součástí práce bude realizace prohlížečky i autorského nástroje.

1. Zmapujte možnosti a existující nástroje pro realizaci interaktivních průvodců webových aplikací.
2. Navrhněte architekturu a koncept fungování takovéhoto řešení s pohledu uživatelské i tvůrčí části. Důraz bude kladen na nezávislost na hostitelské aplikaci a flexibilitu integrace.
3. Implementujte navržené řešení pomocí moderních webových technologií.
4. Zhodnoťte vytvořený nástroj a realizujte případovou studii nasazení a využití systému v praxi.

Seznam doporučené odborné literatury:

[1] Jon Duckett: JavaScript and JQuery: Interactive Front-End Web Development, Wiley, 2014, ISBN: 978-1118531648
[2] Sasha Vodnik: HTML5 and CSS3, Illustrated Complete, Course Technology, 2015, ISBN: 978-1305394049
[3] Jason Beaird: The Principles of Beautiful Web Design, SitePoint, 2014, ISBN: 978-0992279448
[4] Erixc Elliot: Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries, O'Reilly Media, 2014, ISBN: 978-1491950296

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 01.09.2015

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne 7.7.2017

.....

podpis

Tímto bych rád poděkoval vedoucímu práce Ing. Michalu Radeckému, Ph.D. za trpělivost, cenné rady a pomoc při řešení diplomové práce.

Abstrakt

Diplomová práce se zabývá navržením a tvorbou samostatného systému umožňujícího realizaci interaktivních průvodců v rámci obecných webových prezentací. Cíle práce bylo dosaženo navržením architektury a implementací jak části pro tvorbu průvodce, tak části umožňující integraci průvodce s důrazem na obecnou použitelnost algoritmu vyhledání elementu ve stránce i při dílčích úpravách obsahu stránky. Použitelnost programu je demonstrována na konkrétním případu. Hlavním přínosem práce je vytvoření uceleného ekosystému aplikace umožňující tvorbu průvodců, které lze snadno integrovat do jakékoli webové stránky.

Klíčová slova

Algoritmus vyhledání elementu ve stránce, Rozšíření prohlížeče, Webový průvodce

Abstract

Master´s thesis deals with the design and creation of a separate system which allows implementation of interactive guides within general web presentations. The goals of the thesis were achieved by designing the architecture and implementation of both the wizard creation section and the wizard integration section, with an emphasis on the general usability of the element search algorithm in the page with partial page content edits. The applicability of the program is demonstrated on the specific example. The main benefit of this work is to create a comprehensive ecosystem of application that makes guides that can be easily integrated into any web site.

Key words

Element search algorithm in the page, Browser extension, Walkthrough tutorial

Obsah

Seznam použitých symbolů a zkratek

API	-	Application Programming Interface
CSS	-	Cascading Style Sheets
DOM	-	Document Object Model
HTML	-	HyperText Markup Language
JSON	-	JavaScript Object Notation
URL	-	Uniform Resource Locator
WYSIWYG	-	What you see is what you get

Seznam obrázků

1 Úvod

Vzrůstající komplexita webových aplikací klade stále větší a větší nároky na uživatele. Také v případě komplexní webové stránky obsahující mnoho vstupních polí postupuje uživatel při práci se stránkou od jednoho prvku k druhému. Práce s takovýmito aplikacemi není vždy intuitivní, avšak náklady na možné přepracování těchto aplikací jsou příliš vysoké, nebo v krajním případě nejsou takovéto úpravy vůbec možné. Narážíme zde na potřebu usnadnit uživateli používání těchto komplexních systémů. Na tuto potřebu poskytuje odpověď průvodce webovou aplikací, který je předmětem této práce. Klíčovou vlastností je vytvoření průvodce a jeho integrace do stránky, a to bez nutnosti větších zásahů do zdrojového kódu. Takovéhoto průvodce je možno vytvářet pomocí specializovaných programů, které jej umožní přidat do stránky bez nutnosti jej programovat.

Obsahem první části práce je představení některých typických zástupců programů pro vytváření a správu průvodců. V rámci této kapitoly je popsáno, jakou funkcionalitu nabízí a jakým způsobem jsou řešeny klíčové funkce. Ve druhé části práce je popsán návrh a implementace vlastního programového řešení tohoto problému. Výsledkem je pak naimplementované funkční řešení. V poslední části práce je implementovaný program porovnán s programy popsanými v první části a také je provedena případová studie nasazení průvodce na webovou stránku pomocí implementovaného nástroje.

2 Co je průvodce webovou aplikací

Průvodce webovou aplikací je rozšíření stávající webové prezentace o nový prvek, který má uživatelům webové stránky poskytnout nápovědu nebo dodatečnou informaci. Průvodce je složen ze zpráv, které se postupně zobrazují u částí stránky. Typicky bude průvodce použit všude tam, kde je potřeba uživatele webové aplikace:

- **Provést nějakou funkcionalitou** – může se například jednat o složitější formulář, kdy je průvodcem uživatel veden k vyplnění jednotlivých polí formuláře.

- **Představit části stránky** – ve složitějších či uživatelsky neergonomicky navržených webových aplikacích může být vhodné uživateli pomocí průvodce ukázat, jaké funkce či informace může v jednotlivých částech stránky nalézt.

Průvodce je možno vytvářet jak přímo v rámci návrhu a tvorby stránky, tak je možno využít specializovaných programů, které umožní integraci průvodce do již hotových řešení webových aplikací.

3 Existující nástroje pro realizaci interaktivních průvodců webových aplikací

Problém toho, jak umožnit přidat průvodce do existující webové stránky, aniž by bylo nutné jej naprogramovat na úrovni zdrojového kódu, je náplní funkčnosti mnoha softwarových řešení, která jsou na trhu dostupná. Jednotlivé programy nabízejí danou funkcionalitu v různorodé formě. Liší se jak architekturou řešení, tak v množině problémů, které jsou schopná v rámci své funkcionality pokrýt. Představíme si tři programy, které tvoří typické zástupce softwarového řešení tohoto problému. Budou představeny jak programy komerční, tak nekomerční.

3.1 WalkMe

První aplikace, jež bude prezentována v této kapitole, se jmenuje WalkMe[1]. Jedná se o komerční nástroj a pro základní použití, které je funkcionálně omezeno na maximálně pět kroků v jednom průvodci, je možno jej použít zdarma. Pro plné využití je ovšem nutné zakoupení licence. Jedná se o program určený pro tvorbu funkcionálně složitějších průvodců. Umožňuje ve velké míře přizpůsobit jak vzhled, tak průběh spuštění průvodce.

3.1.1 Funkčnost

Aplikace WalkMe je distribuována jako aplikace rozšíření prohlížeče Firefox. Po instalaci je do nástrojové lišty prohlížeče přidána ikona spouštějící aplikaci. WalkMe umožňuje vytvářet průvodce, jehož jednotlivé kroky jsou buď ve formě zprávy, nebo ve formě modálního okna. Program nabízí velké množství funkcí a nastavení. Níže je uveden seznam některých z nich:

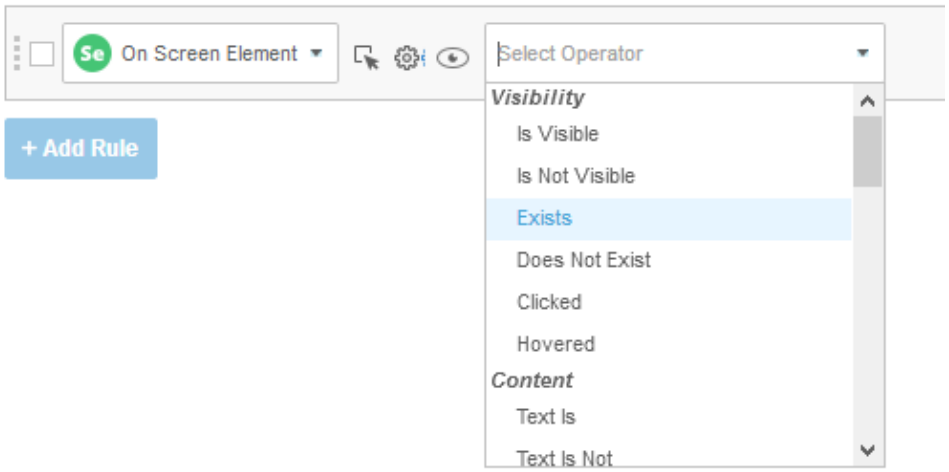
- Chování průvodce je možno ovlivnit sestavením složitých podmínek.
- Kroky průvodce mohou být tvořeny několika různými prvky uživatelského rozhraní.
- Lze definovat selektor pro nalezení elementu zprávy.
- Spuštění průvodce lze regulovat definováním pravidel.
- Možnost definování podmínek, za jakých budou jednotlivé kroky spouštěny.
- Nabízí možnost větvení průběhu průvodce na základě definovaných podmínek.
- Aplikace má velké množství nastavení vzhledu a vlastností zpráv průvodce.
- Aplikace nabízí rozhraní, jenž řeší, který průvodce má být na stránce dostupný.
- Editace zprávy je možná pomocí WYSIWYG editoru.

[1] Webová prezentace výrobce softwaru je dostupná na adrese https://www.walkme.com.

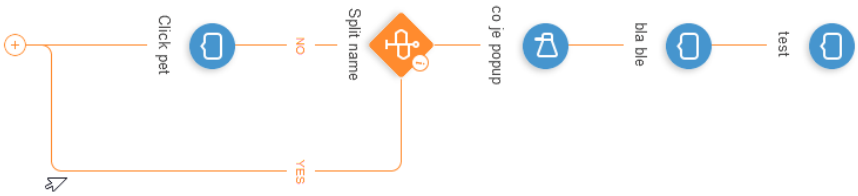
Jednotlivým krokům zprávy je možno určit, za jakých podmínek budou spuštěny. K tomu je v aplikaci k dispozici editor, v němž je možno vytvářet podmínky na základě:

- stavu některého z elementů stránky,
- obsahu URL adresy,
- obsahu dat (Cookies[2], proměnné ve stránce),
- prohlížeče (typu, velikosti stránky),
- data a času.

Výsledkem je pak možnost vytvářet větvení běhu průvodce.

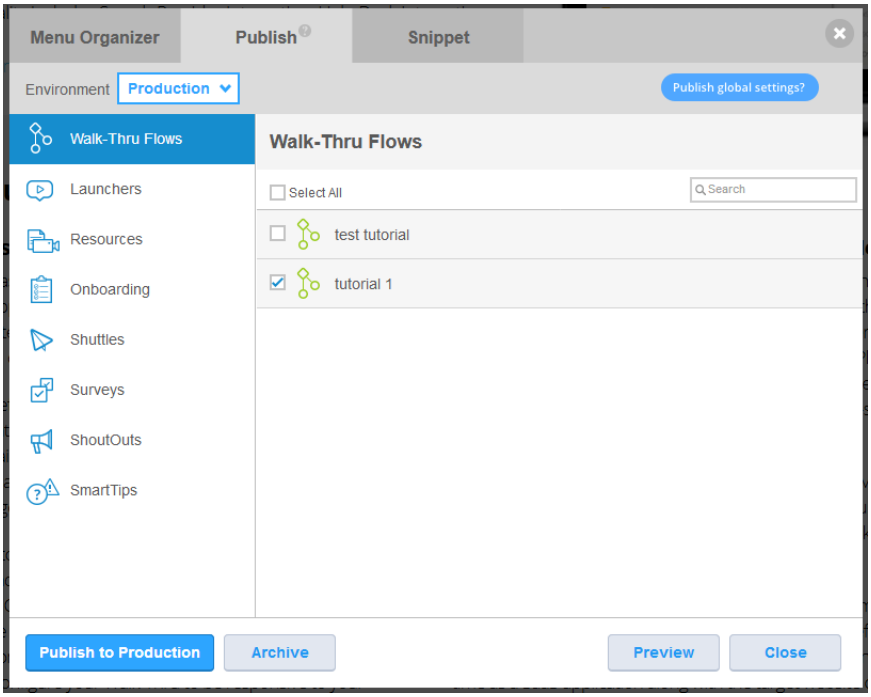


Obrázek 1: Editor pro zadání podmínky



Obrázek 2: Větvení běhu průvodce na základě podmínky

[2] Cookies je malé množství dat, které WWW server pošle prohlížeči a ten je uloží v lokálním úložišti.

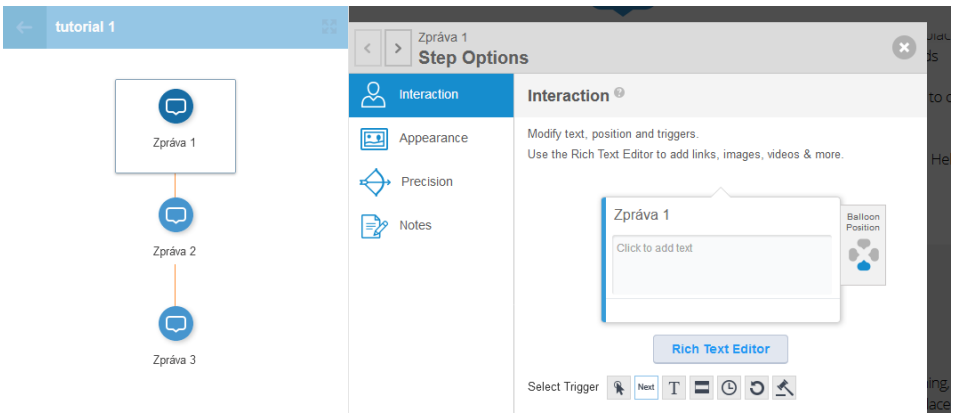


Obrázek 3: Uživatelské rozhraní umožňující nastavit, který průvodce je ve stránce aktivní

Aplikace také umožňuje do stránky integrovat další funkcionalitu jako například live chat, dotazník nebo nástroj pro přehrání multimediálních souborů.

3.1.2 Část aplikace pro tvorbu průvodce

Uživatelské rozhraní pro tvorbu a editaci průvodce je po spuštění aplikace zobrazeno v levé části okna prohlížeče. Uživatelské rozhraní aplikace není součástí struktury webové stránky. Vybírání elementu ve stránce probíhá tak, že při přechodu myši přes element stránky jsou tomuto elementu zvýrazněny okraje. Stejně jako v případě uživatelského prostředí není ani toto zvýraznění řešeno změnami CSS tříd v samotné stránce.



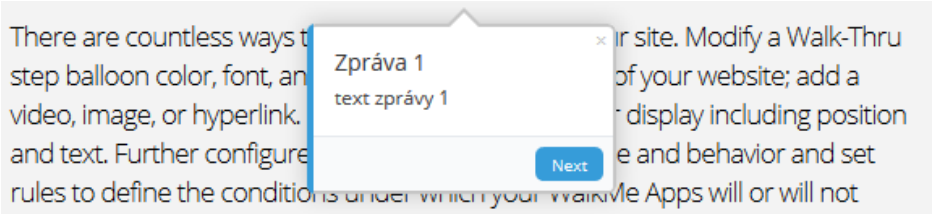
Obrázek 4: Uživatelské rozhraní pro tvorbu průvodců

3.1.3 Integrace průvodce

Integrace průvodce do stránky je řešena formou skriptu, který je potřeba přidat do zdrojového HTML kódu stránky. Z obsahu tohoto skriptu je možno vyčíst, že sám o sobě žádnou funkcionalitu pro vykreslení zpráv, ani samotná data průvodce neobsahuje. Vše je stáhnuto ze serveru poté, co je skript spuštěn.

```
<script type="text/javascript">(function() {var walkme = document.createElement('script'); walkme.type =
'text/javascript'; walkme.async = true; walkme.src = 'https://cdn.walkme.com/users
/f7da81b8060e43c1801578ae2eed5e10/test/walkme_f7da81b8060e43c1801578ae2eed5e10_https.js'; var s =
document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(walkme, s); window._walkmeConfig =
{smartLoad:true}; })();</script>
```

Obrázek 5: Skript aplikace WalkMe spouštějící průvodce



Obrázek 6: Zpráva průvodce v aplikaci WalkMe

3.2 Inline manual

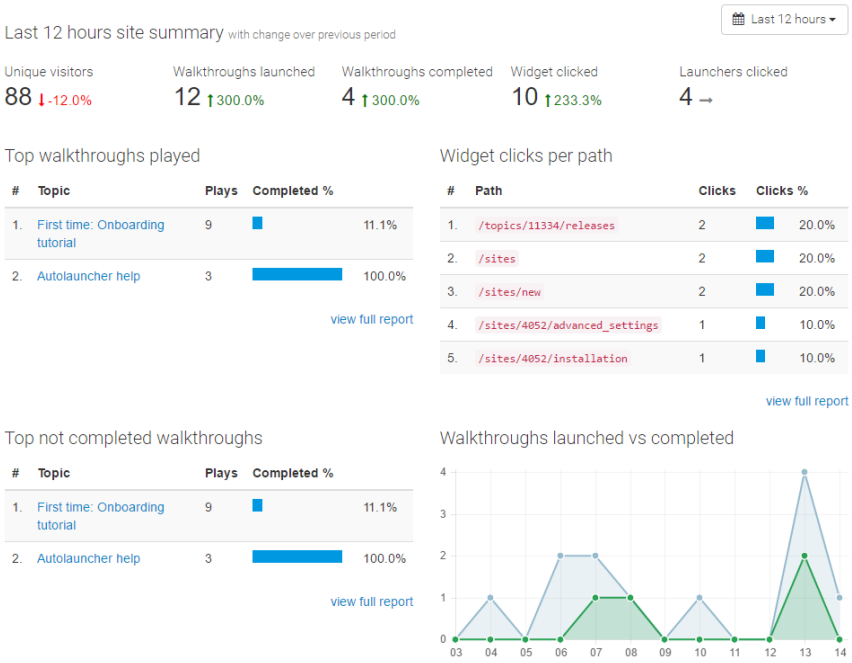
Jedná se o ucelené řešení pro tvorbu a integraci průvodců, tooltipů či oznámení do webové stránky. Jedná se komerční program, jenž je pro testovací účely možné po omezenou dobu využívat zdarma. Po uplynutí této lhůty je pro jeho užívání nutné zaplatit plnou verzi programu.[3]

3.2.1 Funkčnost

Systém Inline manual je distribuován jako aplikace rozšíření prohlížeče Chrome[4]. Po spuštění aplikace se ve stránce v aktuálně zobrazené kartě prohlížeče zobrazí menu s veškerou funkcionalitou pro tvorbu a editaci. Po instalaci rozšíření je do prohlížeče Chrome přidána ikona aplikace. Pokliknutím na tuto ikonu je v levé části aktuálně otevřené stránky zobrazeno uživatelské rozhraní tvůrčího nástroje aplikace. Přestože má uživatelské prostředí velké množství možností, tak je intuitivní a je přehledně rozděleno do sekcí. Aplikace nabízí velké množství funkcí a nastavení vlastností průvodce, přičemž níže jsou uvedeny některé z nich:

[3] Webová prezentace výrobce softwaru je dostupná na adrese https://inlinemanual.com.
[4] Rozšíření prohlížeče jsou aplikace, které jsou instalovány a spouštěny přímo v prohlížeči.

- Kromě textu mohou být obsahem zprávy HTML elementy nebo multimediální obsah,
- možnost volby umístění zprávy,
- tvorba tooltipů,
- tvorbu článků,
- historie úprav,
- analýza používání průvodců uživateli webu,
- možnost vložit do stránky tlačítko, které spustí průvodce,
- editace zprávy pomocí WYSIWYG editoru.

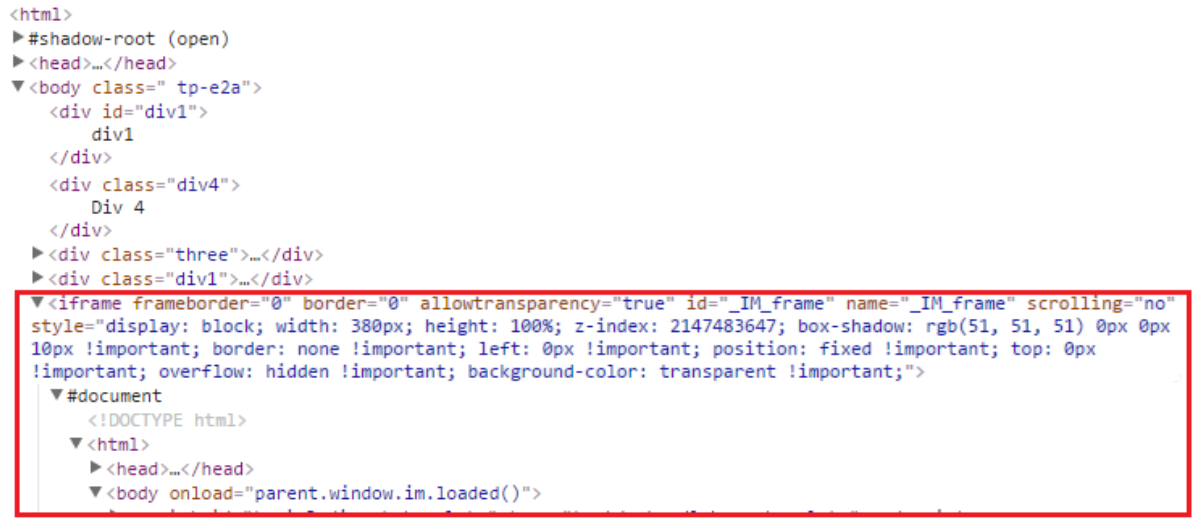


Obrázek 7: Ukázka nástroje pro analýzu návštěvnosti

3.2.2 Část aplikace pro tvorbu průvodce

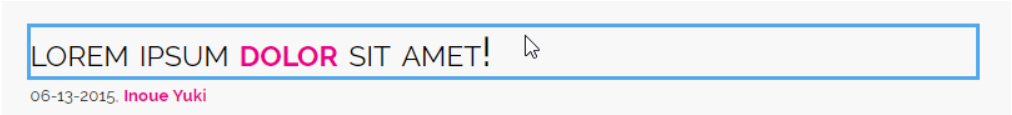
Poté, co je aplikace nainstalována, je do nástrojové lišty prohlížeče Chrome přidána ikona aplikace. Aplikaci pro tvorbu je možno ve stránce spustit zmáčknutím této ikony. Oproti řešení, které používá aplikace Walk Me, je zde rozdíl v tom, jakým způsobem je vykresleno uživatelské rozhraní. V tomto případě je vykreslováno přímo do struktury stránky, se kterou je aktuálně pracováno, a to ve vlastním `iframe`[5] elementu.

[5] Iframe element slouží pro definování plochy ve webové stránce kde je vložena jiná webové stránka.



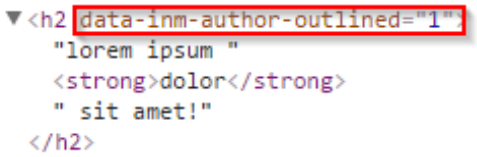
Obrázek 8: Část ve struktuře stránky tvořící uživatelské rozhraní aplikace

Vybírání elementu probíhá ve stránce tím způsobem, že je při pohybu myší zvýrazněn element, který je na nejnižší pozici.



Obrázek 9: Zvýraznění elementu při vybírání

Zvýraznění je technicky řešeno tím, že je elementu přidán atribut, který změní jeho CSS vlastnost `outline`[6]. Tento atribut je elementu dynamicky přidán a poté, co je kurzor přesunut na pozici jiného elementu, je odebrán.



Obrázek 10: Třída přidaná elementu

[6] CSS vlastnost Outline vykresluje kolem elementu čáru.

3.2.3 Integrace průvodce

Spuštění průvodce ve stránce je stejně jako v případě programu WalkMe vyřešeno pomocí skriptu vkládaného do HTML kódu stránky. Skript neobsahuje logiku ani data průvodce. Znamená to, že obojí je stahováno ze serveru výrobce aplikace.

```
<script>!function(){var e=document.createElement("script"),t=document.getElementsByTagName("script")[0];e.async
=1,e.src="https://inlinemanual.com/embed/player.97d4b33dd31846b3d31961ed6c907a74.js",e.charset="UTF-8",t.parent
Node.insertBefore(e,t)}();</script>
```

Obrázek 11: Skript aplikace Inline manual spouštějící průvodce

3.3 WalkHub

Projekt WalkHub je funkcionálně kombinací webové aplikace a skriptu. Projekt je vyvíjen jako open-source pod licencí GNU Affero General Public License ve verzi 3.0. Tato licence opravňuje k tomu, že je možno program bezplatně používat pro nekomerční i komerční účely a také je možno jej libovolně modifikovat. Oproti ostatním nástrojům popisovaných v této kapitole umožňuje WalkHub jen tvorbu základního průvodce. Nenabízí žádné možnosti řízení běhu průvodce nebo využití jiných vizuálních komponent, než jsou jednoduché zprávy. Výhodou ovšem je, že program lze použít zcela zdarma také ke komerčnímu využití.

3.3.1 Část aplikace pro tvorbu průvodce

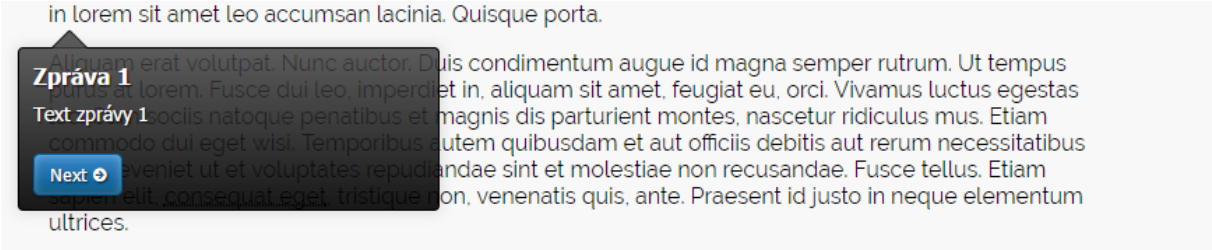
Část aplikace umožňující výběr elementů ve stránce je řešena pomocí skriptu, který je před samotnou tvorbou průvodce zapotřebí vložit do stránky. Součástí skriptu je tlačítko pro spuštění tvorby. Výběr elementů ve stránce probíhá velice podobně jako v případě programu Inline Manual a WalkMe. Při procházení stránkou je zvýrazněn element, nad kterým se nachází kurzor. Toto zvýraznění je řešeno přidáním třídy elementu upravující CSS vlastnost jeho okraje. V samotné stránce probíhá jen výběr elementů, editace obsahu zpráv probíhá na webových stránkách projektu.

```
<script src="https://walkhub.net/assets/client.js" defer></script>
<script src="https://walkhub.net/assets/embed.js" defer></script>
<a class="walkthroughbutton" data-origin="https://walkhub.net/" data-search="" href="https://walkhub.net/"></a>
<a class="walkthroughbutton" data-origin="https://walkhub.net/" href="https://walkhub.net/"></a>
```

Obrázek 12: Skript vygenerovaný aplikací WalkHub spouštějící tvorbu průvodce ve stránce

3.3.2 Integrace průvodce

Integrace průvodce do stránky je vyřešena stejným způsobem jako u ostatních programů. Z webového prostředí aplikace je nutné stáhnout skript a ten do stránky vložit. Po spuštění stránky je v dolním rohu zobrazeno tlačítko umožňující průvodce spustit. Skript opět neobsahuje data ani logiku průvodce.



Obrázek 13: Zpráva průvodce v aplikaci WalkHub

3.4 Shrnutí

V kapitole byly představeny tři programy pro tvorbu průvodců. Jednotlivé programy se liší jak nabízenou funkcionalitou, tak způsobem, jakým je řešeno vykreslení uživatelského rozhraní pro tvorbu průvodce. Programy Inline Manual a WalkMe jsou komerčními projekty a oproti open source programu WalkHub nabízejí přehlednější uživatelské prostředí a mnohem větší možnosti řízení běhu průvodce a nastavení vzhledu. Integrace vytvořeného průvodce do stránek je u všech tří programů řešena stejně. Je vygenerován malý skript, který je nutné vložit do stránky ručně. Veškerá funkcionalita i data potřebná pro vykreslení zpráv ve stránce jsou stažena ze serveru.

4 Koncept funkčnosti a návrh architektury vyvíjeného průvodce webovými aplikacemi

Návrh aplikace je klíčovou součástí jejího vývoje a je nutné se před tím, než začneme aplikaci implementovat, rozmyslet, jaká funkcionalita je požadována a jak jí je možno co nejlépe docílit.

Cílem praktické části práce je vyvinout systém, který umožní vytvořit průvodce pro libovolnou webovou stránku a bude poskytovat možnosti, jak jej do stránky snadno integrovat. Pro to, aby systém mohl takovouto funkcionalitu poskytnout, je nutno položit si následující otázky a v rámci této kapitoly na ně nalézt odpovědi:

- Jak bude probíhat tvorba průvodce?
- Jak umožnit výběr elementů ve stránce?
- Jak a kde poběží uživatelské prostředí aplikace?
- Jakým způsobem vybraný element uložit a přiřadit k němu zprávu?

Aby mohl být průvodce spuštěn ve webové stránce, je nutné navrhnout funkcionalitu, která bude odpovědí na následující otázky:

- Jak vytvořeného průvodce integrovat do stránky?
- Jak bude spuštěný průvodce ve stránce vyhledávat elementy?
- Jak u nalezeného elementu zobrazit zprávu?

Návrh aplikace tedy rozdělíme na dvě části. Na část, která průvodce tvoří, a na část, která průvodce spouští.

4.1 Aplikace pro tvorbu průvodce

Hlavním smyslem tvorby aplikace je umožnit vytvořit a přidat do stránky průvodce uživatelům, kteří neumí, nebo nechtějí programovat průvodce na úrovni zdrojového kódu. Úkolem části aplikace pro tvorbu průvodce je tuto funkcionalitu uživateli nabídnout. Část pro tvorbu bude pomocí uživatelského rozhraní umožňovat vytvořit průvodce, jednotlivé zprávy a jejich obsah a také vybrání elementu ve stránce, u nějž má být zpráva zobrazena. Průvodce bude možné pomocí aplikace editovat. Aplikace musí umožnit vytvořeného průvodce integrovat do webové stránky.

4.1.1 Diagram případů užití

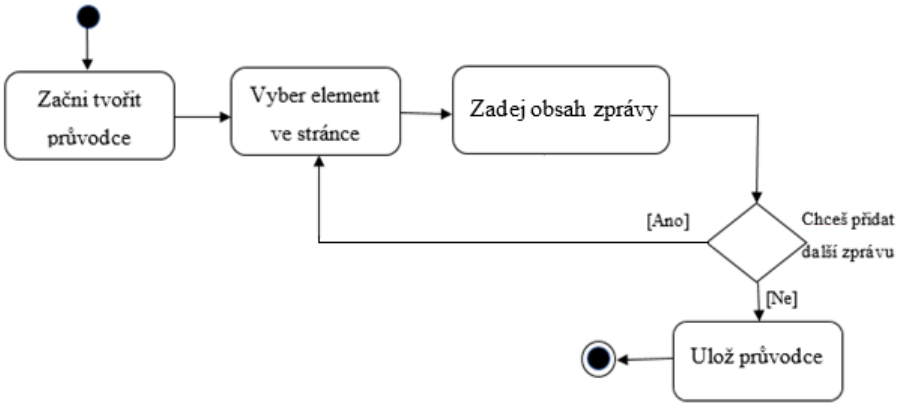
Diagram užití je funkční náhled, který nám pomůže definovat, jací aktéři existují vně vyvíjeného systému a co má být prováděno systémem (případy užití) [1].

Obrázek 14: Diagram případů užití

Jak je vidět z diagramu případů užití na obrázku 14, s částí systému pro tvorbu průvodců bude pracovat jen jeden aktér, a to tvůrce průvodce. Nyní budou představeny jednotlivé případy užití.

Tvorba průvodce

Jedná se o stěžejní funkci systému. Uživatel bude mít možnost tvořit průvodce a přiřazovat mu zprávy. Jednotlivým zprávám bude možné přiřadit HTML element stránky. Zprávám taky bude možné vytvořit obsah. Tvorba průvodce z pohledu uživatele je znázorněna v aktivitním diagramu na obrázku 15.



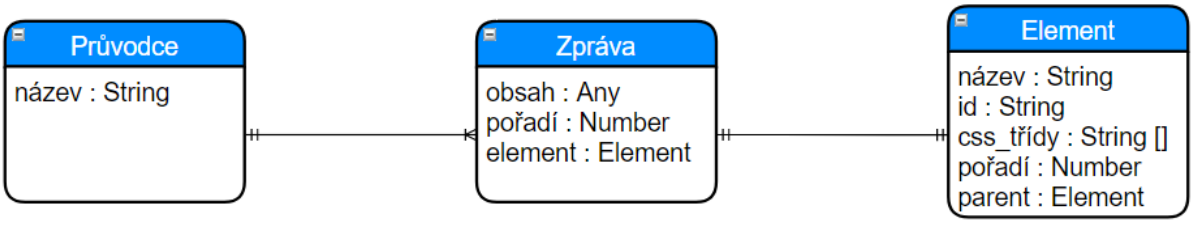
Obrázek 15: Aktivitní diagram tvorby průvodce z pohledu uživatele

Z hlediska pohodlného používání aplikace pro tvorbu průvodce je důležité, aby systém obsahoval následující funkcionalitu:

- **Editace obsahu zprávy** – uživatel bude moct editovat obsah zprávy v již existujícím průvodci.
- **Změna pořadí zprávy** – existence možnosti měnit pořadí zpráv v průvodci.
- **Smazání zprávy** – uživatel bude moci odstranit zprávu z průvodce.
- **Zobrazení elementu zprávy** – sloužící k tomu, aby bylo možné zkontrolovat, jaký element je ke zprávě přiřazen. Systém umožní ve stránce zobrazit element přiřazený ve zprávě.
- **Uložení a načtení průvodce** – systém bude umožňovat vytvořené průvodce uložit do souboru tak, aby bylo případně možné se k tvorbě průvodce vrátit při novém spuštění aplikace.

4.1.2 Datový model průvodce

Datový model popisuje strukturu dat v systému a určuje vzájemné vztahy jednotlivých datových prvků navzájem, čímž v nich reprezentuje vymezenou část reality. Pro popis, jak vypadají datové struktury používané v aplikaci, použijeme ER diagram (Entity relations diagram).



Obrázek 16: ER diagram

Základní používanou entitou v aplikaci je průvodce. Průvodce obsahuje zprávy. Jednotlivé zprávy se skládají z nějaké informace, typicky to bude nadpis a obsah zprávy. V případě rozšíření programu by to ale mohl být například nějaký multimediální obsah či obsah v podobě HTML kódu. Důležitou vlastností zprávy je její pořadí. Pořadí zprávy určuje, jak jsou zprávy za sebou zobrazovány.

Jednotlivé zprávy jsou vždy zobrazeny u některého z elementů stránky. Z tohoto důvodu obsahuje zpráva entitu element. Entita element slouží k uložení vlastností vybraného elementu, včetně struktury vedoucí ke kořenovému elementu stránky.

4.1.3 Vybírání elementu ve stránce

Jednotlivé zprávy průvodce jsou navázány na elementy stránky. Při tvorbě průvodce je tedy zapotřebí umožnit uživateli zvolení elementu zprávy. Z důvodu, že se může struktura stránky během práce s ní měnit, a také proto, že potřebujeme mít při tvorbě průvodce přístup ke stejné struktuře stránky, jako je ta při jeho spuštění, je nutné průvodce vytvářet ve stejném prostředí, v jakém je spouštěn.

Z několika možných řešení byla zvolena cesta přidání skriptu do stránky. Z prostředí skriptu, který běží ve stránce, máme plný přístup ke struktuře spuštěné stránky, což nám umožní uložení celé cesty elementu s jeho identifikátorem, třídami a rodičovským elementem do struktury entity element z ER diagramu. Skript také umožňuje strukturu stránky měnit, díky čemuž lze vybírání elementu graficky vizualizovat přímo ve stránce.

4.1.4 Prostředí uživatelského rozhraní tvorby průvodce

Jedním z kroků, které je třeba v rámci návrhu aplikace udělat, je rozhodnout, v jakém prostředí poběží uživatelské rozhraní. Abychom umožnili výběr elementu, je nutné ve stránce spustit skript. Obsahem tohoto skriptu je logika řešící výběr elementu.

Můžeme využít toho, že do stránky vkládáme skript a spouštíme v ní vlastní funkcionalitu. Vkládaný skript můžeme rozšířit o logiku vykreslující ve stránce uživatelské rozhraní aplikace. Uživatelské rozhraní aplikace tedy bude vykreslováno přímo ve webové stránce, pro kterou je průvodce tvořen. Vkládaný skript bude tedy obsahovat jak logiku řešící výběr elementu ve stránce, tak logiku pro vykreslení uživatelského rozhraní.

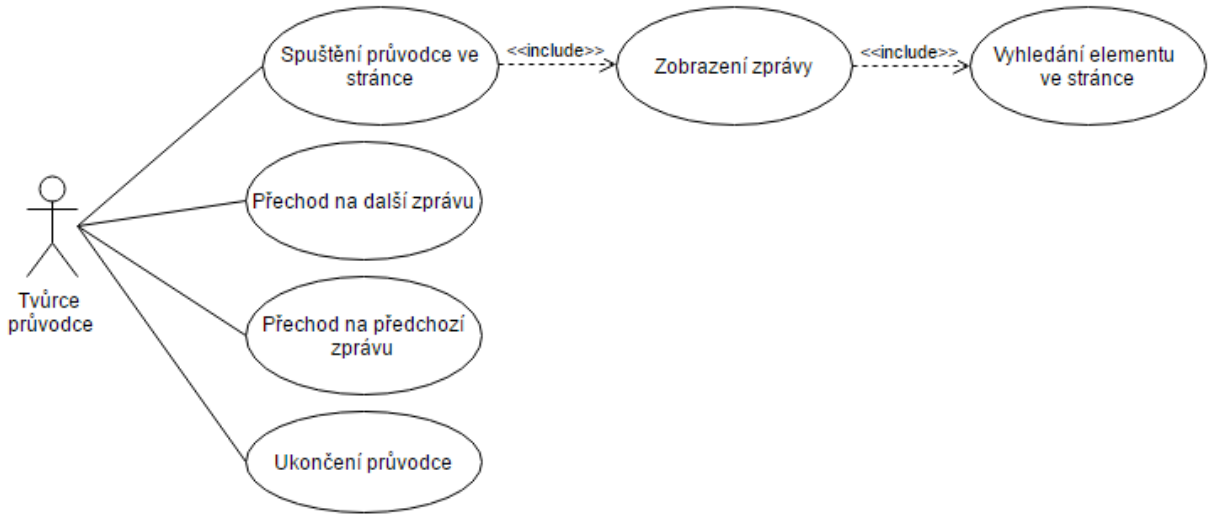
4.2 Aplikace spouštějící průvodce

V předchozích částech bylo navrženo, jaký bude koncept fungování části aplikace pro tvorbu průvodců, ale zatím nebylo řešeno, jakým způsobem bude vytvořený průvodce integrován do stránky. Průvodce bude spuštěn při běžné návštěvě webové stránky uživatelem, který nemusí mít nainstalováno nic jiného než běžný webový prohlížeč. Z tohoto požadavku vyplývá, že průvodce musí být buď součástí stránky, nebo musí stránka obsahovat aparát k tomu, aby si potřebnou funkcionalitu stáhla z jiného umístění. V obou případech je nutné kód stránky nějakým způsobem rozšířit.

Všechny aplikace pro tvorbu průvodců představené v kapitole 3 využívají druhého přístupu. Do stránky je přidán jen krátký skript, který obsahuje instrukce jak pro stáhnutí veškeré funkcionality potřebné pro spuštění průvodce ze serveru a dynamické přidání do stránky, tak i data samotného průvodce.

Výhodou přístupu, kdy je ve zdrojovém kódu stránky umístěn jen krátký skript a vše ostatní je uloženo na serveru, je to, že je pak možné upravovat průvodce, aniž by bylo nutné skript ve stránce měnit. Na druhou stranu výhodou přístupu, kdy je do stránky vložena funkcionalita i data průvodce, je, že bude fungovat i v případě, když budeme stránku spouštět z lokálního umístění bez přístupu k internetu. Dnes je pomocí webových technologií možno psát i desktopové aplikace, které mohou běžet na strojích bez přístupu k internetu. Pro takový případ bude možno použít pouze tento přístup. Aby bylo možné podporovat i řešení, která nemají přístup k internetu, bude v aplikaci zvolen přístup, kdy je veškerá funkcionalita i data součástí skriptu vkládaného do stránky.

4.2.1 Diagram případů užití



Obrázek 17: Diagram případů užití

Jak je vidět z diagramu případů užití na obrázku 17, s částí systému pro spuštění průvodce bude pracovat jen jeden aktér, uživatel webové aplikace. Uživatel webové aplikace je člověk, který v prohlížeči spustí webovou stránku obsahující průvodce. Níže budou ukázány jednotlivé případy užití:

- **Spuštění průvodce ve stránce –** uživatel průvodce spustí kliknutím na tlačítko ve stránce. Poté je zobrazena první zpráva. Zprávy budou zobrazeny vedle elementů, které jim byly při vytvoření vybrány.
- **Přechod na další zprávu a přechod na předchozí zprávu –** mezi jednotlivými zprávami průvodce bude možno procházet dopředu i dozadu.
- **Ukončení průvodce –** ukončit průvodce lze kdykoli pomocí tlačítka v uživatelském rozhraní.

4.2.2 Vyhledání elementu ve stránce

Poté, co je průvodce spuštěn, je třeba ve stránce nalézt první element, u kterého průvodce začíná. V případě, že DOM[7] stránky při spuštění průvodce bude naprosto totožný s tím, jaký byl při jeho vytváření, bude takový úkol triviální. Jelikož má HTML dokument stromovou strukturu, bylo by možné jej jednoduše projít od kořenového elementu a vybíráním stejných potomků, jako byli v původním dokumentu, bychom se postupným procházením stromu dokumentu dostali až k hledanému elementu. V praxi se ovšem může stát, že prohledávaný dokument bude lehce odlišný. To se může stát ze dvou důvodů:

- V době mezi tvorbou tutoriálu a jeho spuštěním byly na stránce provedeny změny programátorem. Typicky se může jednat například o přidání sekce do stránky či odebrání nějaké části.
- Při každém ze spuštění může být stránka lehce odlišná. Příkladem může být například rozdíl ve stránce v případě, kdy je uživatel přihlášen a kdy přihlášen není. Velice často má stránka v takovémto případě rozdílné menu a zobrazení některých částí stránky. Znamená to, že také struktura stránky bude odlišná.

Požadavkem je, aby byl algoritmus schopen element najít i v případě podobných změn v co největším počtu případů.

[7] DOM stránky je zkratkou pro Document Object Model. Je to objektově orientovaná reprezentace HTML dokumentu v prohlížeči.

4.2.2.1 Návrh algoritmu

Změny mezi stránkou při tvorbě průvodce a jejím spuštěním mohou nastat ve:

- **Struktuře** – což znamená, že některé části ve stránce mohou být odebrány, přidány nebo přemístěny.
- **Vlastnostech elementů** – typicky to budou hlavně změny tříd elementů.

Obě tyto potenciální změny byly při návrhu algoritmu pro vyhledání elementu ve stránce brány v potaz. Nyní si představíme pseudokód hlavní části algoritmu.

```
1    // hledaný element
2    let cursor;
3    if hasId (cursor) AND document contains element with cursor id then
4        return get element by ID from document
5    endif
6    let elements = find elements in document with same tag as cursor element tag
7    let longestPrefix = 0
8    for element in elements
9        let prefixLength = get length of common path prefix of element and cursor
10       if prefixLength > longestPrefix then
11           longestPrefix = prefixLength
12       endif
13   for element in elements
14       if get length of common path prefix of element and cursor != longestPrefix then
15           remove element from elements
16       endif
17
18   if sizeof(elements) == 0 then
19       return NOT FOUND
20   elseif sizeof(elements) == 1 then
21       return elements[0]
22   elseif sizeof(elements) > 1 then
23       let highestScore = 0
24       let highestScoredElement = NOT FOUND
25       for element in elements
26           if child position heuristic (element) == 1 then
27               return element
28           endif
29           let score = child position heuristic (element)
30                       + classes startegy heuristic (element)
31           if score > highestScore then
32               highestScore = score
33               highestScoredElement = element
34           endif
35       return highestScoredElement
```

Obrázek 18: Pseudokód hlavní větve algoritmu

Algoritmus je založen na tom, že se v případě, že vyhledávaný element nemá identifikátor, nebo element s takovým identifikátorem nebyl ve stránce nalezen, snažíme ve stránce najít element, který má

co nejdelší stejnou cestu ke kořenovému elementu, jako je zaznamenáno pro vyhledávaný element. To znamená, že v první fázi výběru se zaměřujeme jen na podobnost ve struktuře dokumentu a kromě typu nebereme v potaz jiné vlastnosti jednotlivých elementů, jako jsou třídy nebo pořadí potomka vůči rodiči. Pokud je takovýchto elementů nalezeno více, porovnáváme jednotlivé kandidáty na co největší shodu. Pro hledání shody je využíváno dvou různých strategií:

- podobnost v pozici potomka vůči předkovi,
- podobnost CSS tříd elementů.

Jak je vidět v pseudokódu hlavní větve algoritmu, je v tomto případě preferována shoda na pozici potomka, ale díky tomu, že strategie jsou odděleny od hlavní větve programu, je možno velice jednoduše měnit, na co chceme při vyhledávání dávat větší důraz. Obě tyto strategie jsou popsány pomocí pseudokódu na obrázcích 20 a 21.

Rozhraní strategie obsahuje jedinou funkci, jejíž parametry jsou hledaný element a vyhodnocovaný element. Návratovou hodnotou je reálné číslo v rozmezí 0 až 1, přičemž 0 znamená žádnou shodu a 1 shodu dokonalou.

```
1 Strategy : getScore ( target, evaluated ) : Real
```

Obrázek 19: Pseudokód rozhraní strategie

```
1   // hledany element
2   let cursor;
3   // vyhodnocovany element
4   let candidate;
5   let diffSum = 0;
6   while cursor not null and candidate not null
7     diffSum += position(cursor) - position(candidate)
8     cursor = parent(cursor)
9     candidate = parent(candidate)
10  endwhile
11
12  if diffSum == 0 return 1
13  else return 1 / (1 + diffSum)
```

Obrázek 20: Pseudokód strategie podobnosti podle pořadí potomka

```
// hledany element
    let cursor;
    // vyhodnocovany element
    let candidate;
    let diffSum = 0;
    while cursor not null and candidate not null
        diffSum += classesDiff(cursor, candidate)
        cursor = parent(cursor)
        candidate = parent(candidate)
    endwhile

    if diffSum == 0 return 1
    else return 1 / (1 + diffSum)


    function classesDiff(cursor, candidate)
       Set cursorClasses = cssClasses(cursor)
       Set candidateClasses = cssClasses(candidate)
       let total = union(cursorClasses , candidateClasses)
       let intersecting = intersection(cursoClasses, candidateClasses)
       return total - intersecting
```

Obrázek 21: Pseudokód strategie podobnosti tříd

4.2.2.2 Složitost algoritmu

V této kapitole si rozebereme časové složitosti jednotlivých částí algoritmu. Začneme strategiemi, neboť ty jsou využívány v hlavní větvi programu.

Strategie podle podobnosti pozice potomka

- Velikost vstupu algoritmu je definována jako minimum délek cesty (od kořene DOMu) ohodnocovaného elementu a délky cesty hledaného elementu.
- Funkce `position` a `parent` mají konstantní složitost O (1).
- Algoritmus obsahuje jeden `while` cyklus.

Má tedy lineární složitost vzhledem k velikosti vstupu.

Strategie podle podobnosti tříd

- Funkce `classesDiff` má lineární složitost O (t + c) vzhledem k množství CSS tříd u kandidáta (c) a množství CSS tříd u hledaného prvku (t), protože množinové operace sjednocení a průniku mají lineární složitost.
- Algoritmus obsahuje jeden `while` cyklus, na svém vstupu. Ve `while` cyklu dochází jednou k volání funkce `classesDiff`.

- Velikost vstupu algoritmu je tedy definována jako minimum délek cesty (od kořene DOMu) ohodnocovaného elementu a délky cesty hledaného elementu a součet CSS tříd v každé úrovni zanoření.

  Na tomto vstupu má z důvodů zmíněných výše algoritmus lineární složitost.

Hlavní větev algoritmu

- Získání elementu z DOMu pomocí ID má složitost O(1) (implementace v prohlížečích).
- Získání elementů z DOMu pomocí názvu elementu má složitost O(1) nebo O(domSize) (podle implementace v prohlížečích), kde domSize reprezentuje počet elementů v DOMu stránky.
- Získání délky společného prefixu cesty má lineární složitost vzhledem k velikosti společné cesty.
- Složitost jednotlivých strategií byla rozebrána výše.
- Dále algoritmus obsahuje dva `for` cykly, které proběhnou vždy, a jeden `for` cyklus volající strategie, který proběhne, pokud je potřeba.

Vzhledem k tomu, že všechny části algoritmu mají lineární nebo konstantní složitost a v algoritmu nejsou vnořené smyčky, spadá i celý algoritmus do lineární třídy složitosti. Velikost vstupu algoritmu je pak definována množstvím elementů ve stránce.

4.3 Shrnutí

Výsledkem fáze návrhu je rozhodnutí, že aplikace pro tvorbu bude implementována jako skript, který po spuštění ve webové stránce prohlížeče zobrazí veškeré uživatelské rozhraní ve stránce. Tvorba průvodce a veškerá další práce s ním tedy bude probíhat přímo v okně prohlížeče, což nám umožní přístup ke struktuře stránky, aby bylo možné naimplementovat funkcionalitu pro výběr elementu.

Aplikace pro spuštění průvodce bude také naimplementována jako skript. Ten bude možné stáhnout z uživatelského rozhraní aplikace pro tvorbu. Bylo rozhodnuto, že data nutná pro spuštění průvodce budou součástí tohoto skriptu. V rámci návrhu této části aplikace byl představen algoritmus umožňující vyhledání elementu ve stránce, který je popsán pseudokódem, a byla vypočítána jeho časová složitost.

Jedním z cílů práce je, aby spuštění průvodce bylo nezávislé na prohlížeči. To je zajištěno již návrhem aplikace tím, že spuštění průvodce je realizováno samostatným skriptem.

5 Implementace

V kapitole Implementace navážeme na rozhodnutí, která byla udělána v návrhu. Bude vysvětleno, jaké nástroje byly pro generování skriptů použity, jaké jsou implementační detaily stěžejních funkcí a jakým způsobem jsou přenášena data z části pro tvorbu do části pro spuštění. Bude také představeno uživatelské rozhraní obou částí.

5.1 Volba programovacího jazyka

Jelikož je jak část pro tvorbu, tak část pro spouštění vyvíjena jako skripty spouštěné v prohlížeči ve stránce, musí být jazykem skriptu JavaScript. A to z důvodu, že je jako jediný klientský skriptovací jazyk podporován všemi prohlížeči. Skript ovšem není nutné psát přímo v jazyce JavaScript. Je možné využít některý z jiných programovacích jazyků, jako je TypeScript, nebo Elm[8], který je do JavaScriptu transpilován[9].

TypeScript je open source programovací jazyk vyvíjený a spravovaný firmou Microsoft. Jedná se o striktní syntaktickou nadmnožinu jazyka JavaScript. To znamená, že jakýkoli kód napsaný v Jazyce JavaScript je také validním kódem v jazyce TypeScript. TypeScript kód je transpilován do JavaScript soboru. TypeScript oproti čistému JavaScriptu přidává například:

- podporu typování,
- podporu genericity,
- psaní tříd a dědičností stylem podobným programovacímu jazyku C#,
- bezpečnější refaktorování.

TypeScript také podporuje takzvané source map definition files. Jedná se v podstatě o hlavičkové soubory, které rozšiřují existující knihovny o typové informace. To usnadňuje použití externích knihoven jako například JQuery při psaní kódu v Typescriptu, neboť nápověda pro takovou externí funkci je pak zobrazena, jako by knihovna byla psána v jazyce Typescript. Prohlížeč využívá source map definition files, aby bylo možno vizuálně ladit kód v TypeScript souboru, i když vnitřně v prohlížeči běží jako JavaScript [9].

[8] Elm je funkcionální silně typovaný programovací jazyk, transpilovaný do JavaScriptu. Více informací na https://guide.elm-lang.org.
[9] Transpilace je proces, kdy je zdrojový kód jednoho jazyka přeložen do zdrojového kódu jazyka jiného.

```
abstract class Component {
    protected _id: string;

    get id(){
        return this._id;
    }

    children: Component[] = [];

    constructor() {
        this._id = Helpers.generateGuid();
    }

    addComponentToDOM(): void {
        Helpers.addHtmlToElementEnd('myTutorial', this.getComponentsHtml());
        this.activateEventHandlers();
    }

    removeComponentFromDOM(): void {
        let element = document.getElementById(this.id);
        element.remove();
    }

    activateEventHandlers(): void {
        this.children.forEach(child => child.activateEventHandlers());
    }

    abstract getComponentsHtml(): string;
}
```

Obrázek 22: Ukázka kódu aplikace napsaného v jazyce TypeScript

Hlavně z důvodů typové podpory a možnosti snadněji psát třídy a dědičnosti bude kód aplikace psán v programovacím jazyce TypeScript.

Transpilace TypeScript souborů do jednoho JavaScript souboru je prováděna nástrojem Bobril build10. Bobril build je založený na Node.js11 a slouží k vytváření single page aplikací psaných v jazyce JavaScript s mnoha optimalizacemi pro komponentově orientovaný framework Bobril, který je inspirován frameworkem React [3]. Transpilovaný JavaScript soubor je také BobrilBuildem zminifikován. Minifikace je proces, jehož výsledkem je kód se stejnou funkcionalitou, ale je uložen na jeden řádek za použití co nejmenšího počtu znaků. Jsou odstraněny komentáře, nedostupné části kódu a jsou přejmenovány proměnné.

[10] Více informací o BobrilBuildu na www.codeproject.com/Articles/1167901/Bobril-V-Bobril-build.
[11] Node.js je softwarový systém navržený pro psaní vysoce škálovatelných internetových aplikací. Více informací na https://nodejs.org.

5.2 Implementace skriptu pro tvorbu

Ve fázi návrhu bylo rozhodnuto, že část aplikace, pomocí které bude průvodce tvořen, bude implementována jako skript. V kapitole je popsáno, jakým způsobem je skript vkládán do stránky. Další část je pak věnována představení funkcionality výběru elementu ve stránce a v poslední části je ukázáno, jak je ve stránce vykreslováno uživatelské rozhraní, a jsou představeny jeho jednotlivé části.

5.2.1 Vložení skriptu do stránky

Ve fázi návrhu jsme rozhodli, že skript pro tvorbu musí běžet ve stránce prohlížeče. Skript lze do stránky vložit dvěma způsoby.

- Skript je přidán do stránky manuálně ještě předtím, než je spuštěna v prohlížeči.
- Skript je do stránky vložen pomocí rozšíření prohlížeče.

Prohlížeče umožňují naistalovat multiplatformní zásuvné moduly známé jako rozšíření prohlížeče (dále jen rozšíření). Nainstalované rozšíření umožňuje vložit skript do běžící stránky. Použití rozšíření nám oproti manuálnímu vložení skriptu do stránky přinese tři výhody:

- Spuštění aplikace probíhá z prostředí prohlížeče, v jehož nástrojové liště je zobrazena ikona aplikace.
- Tvorba průvodce nevyžaduje, abychom upravovali zdrojový kód stránky, proto není nutné mít přístup k souboru stránky na serveru a upravovat stránku manuálně.
- Třetím benefitem tohoto přístupu je, že aplikace může být distribuována jako rozšíření a je možno ji nainstalovat přímo v uživatelském rozhraní prohlížeče, nebo ze stránek internetového obchodu prohlížeče Chrome[12].

Vzhledem k popsaným výhodám je část aplikace pro tvorbu implementována jako rozšíření prohlížeče.

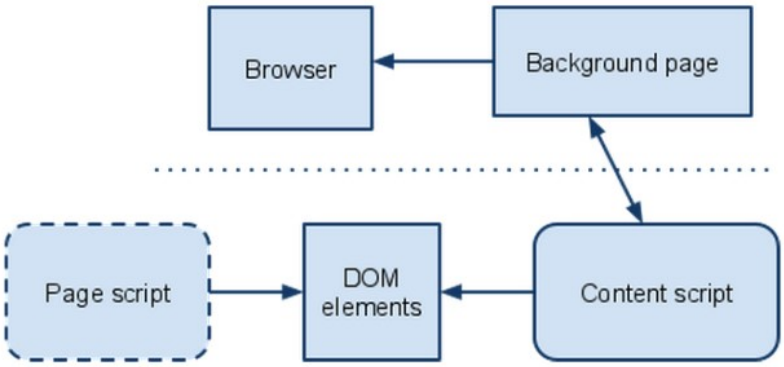
[12] Dostupné na adrese https://chrome.google.com/webstore/category/extensions?hl=cs.

5.2.1.1 Rozšíření prohlížeče

Rozšíření jsou principiálně nezávislá na konkrétním prohlížeči, neboť všechny hlavní prohlížeče (Google Chrome, Opera, Mozilla Firefox, Edge) pro jejich vývoj podporují stejné API. V tom, které funkce z API již prohlížeče implementovali a podporují, se ovšem liší[13]. Je to podobná situace jako s podporou kaskádových stylů ve verzi tři v jednotlivých prohlížečích[14].

Z důvodu, že ne všechny prohlížeče podporují API rozšíření ve stejné míře, je implementace aplikace cílena na prohlížeč Google Chrome, který je na trhu majoritním prohlížečem[15]. V prohlížeči Google Chrome se takové aplikace nazývají Chrome extensions.

Chrome extensions běží uvnitř prohlížeče Chrome a mohou poskytovat dodatečnou funkcionalitu oproti běžným funkcím prohlížeče. Jsou psány pomocí běžných technologií pro webový vývoj, jako je jazyk JavaScript, značkovací jazyk HTML a kaskádové styly.



Obrázek 23: Architektura aplikace rozšíření[16]

Na obrázku 23 je zobrazeno, jakým způsobem aplikace rozšíření funguje. V prohlížeči, ve kterém je nainstalováno rozšíření, je automaticky spuštěn skript event page daného rozšíření. Pokud má tento skript nějakou práci, běží v pozadí ve skryté stránce, takzvané Background page. Event page může reagovat na systémové události, vložit do stránky skript a případně s tímto skriptem komunikovat pomocí zpráv. Vložený skript (takzvaný content script) již může fyzicky měnit strukturu (DOM) stránky [4].

---

[13] Jaká je podpora rozšíření v jednotlivých prohlížečích převzato ze stránky https://developer.mozilla.org/en-US/Add-ons/WebExtensions/Browser_support_for_JavaScript_APIs.
[14] Jaká je podpora vlastností CSS3 v jednotlivých prohlížečích lze nalézt na stránce https://www.w3schools.com/cssref/css3_browsersupport.asp.
[15] Data o rozšíření jednotlivých prohlížečů získána ze stránky https://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=0 dne 29.6.2017.
[16] Obrázek převzat z https://doancphong.wordpress.com/2010/09/22/architecture-model-of-chrome-extension.

V aplikaci je Event page skript v rozšíření využit pro hlídání, zda je zmáčknuta ikona aplikace v prohlížeči. V případě, že zmáčknuta je, je ikona změněna na aktivní a do stránky v aktuálně zobrazené záložce je nahrán content script. V našem případě content script představuje skript pro tvorbu a editaci průvodců. Content script ovšem běží v izolovaném prostředí oproti běžným skriptům stránky. Nemá tedy přímý přístup k proměnným a funkcím skriptů stránky. Stejně tak skripty stránky nemají přístup ke Content scriptu.

```
chrome.pageAction.onClicked.addListener(function (tab) {
    chrome.pageAction.setIcon({ tabId: tab.id, path: "icons/iconEnabled.png" });
    chrome.tabs.executeScript({
        file: 'contentScript.js'
    });
    chrome.tabs.insertCSS({
        file: 'contentScript.css'
    });
});
```

Obrázek 24: Část kódu v event page skriptu, který hlídá, zda byla ikona aplikace zmáčknuta.

Rozšíření nepředstavuje klíčový architektonický prvek aplikace a skript by bylo možné do stránky vložit také ručně. Rozšíření je použito kvůli výhodám uvedených na začátku kapitoly.

5.2.2 Vybírání elementu ve stránce

Během tvorby zprávy průvodce je nutné ve stránce vybrat element. Pro způsob, jak vizualizovat vybírání elementu, jsme se inspirovali řešením, které používají prohlížeče pro výběr elementů v rámci svých vývojových nástrojů. Výběr elementů tam funguje tím způsobem, že je ve stránce zvýrazněn element, který se aktuálně nachází na nejnižší pozici pod kurzorem myši. Takovéhoto chování je možno docílit programově v JavaScript kódu.

Prohlížeč Chrome nenabízí samostatnou funkci pro reakci sledování pohybu kurzoru myši ve stránce. Aby bylo možno reagovat na událost změny polohy myši, je nutno reakci na tuto událost přiřadit některému z elementů ve stránce. Z tohoto důvodu je technické řešení takovéto:

Vložení elementu přes celou stránku

Do stránky je přidán HTML element DIV (dále překrývající element), který je roztažen přes celou stránku. Toho je docíleno tím, že má CSS vlastnost `position` nastavenu na hodnotu `fixed` a CSS vlastnosti `width` a `height` na hodnotu 100 %.

Pomocí CSS vlastnosti `background` má tento překrývající element nastaveno průhledné pozadí, aby šla stránka pod ním vidět tak, jako by tam nebyl.

Reakce na událost pohybu myši

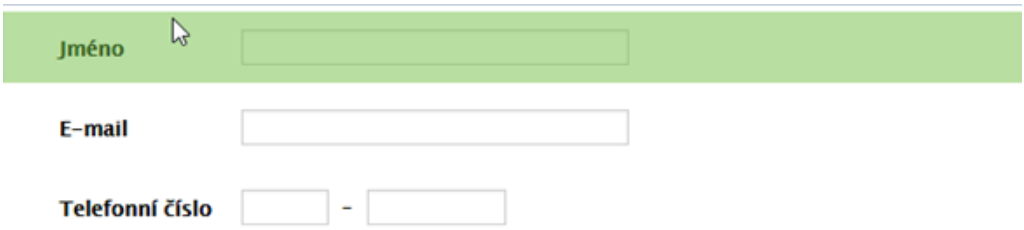
Elementy se mohou ve stránce překrývat, přičemž událost pohybu kurzoru myši je zachycena vždy jen nejvrchnějším elementem. Z tohoto důvodu má překrývající element nastavenu CSS vlastnost `z-index`[17] na maximální hodnotu, aby byl ve stránce umístěn nad všemi ostatními elementy. Překrývajícímu elementu je pomocí JavaScript funkce `AddEventListener` přiřazena reakce na událost pohybu myši[18]. Tato událost vrací hodnotu aktuální polohy kurzoru myši ve stránce.

Hledání elementu na daném umístění

Pomocí JavaScript funkce `elementFromPoint`[19] zjistíme, jaký element je na pozici, který je umístěn nejvýše. Nejvýše je námi přidaný element DIV a z toho důvodu je nutné, než použijeme funkci `elementFromPoint`, změnit CSS vlastnost překrývajícího elementu tak, aby pro danou chvíli nebyl ve stránce vykreslen. Z množství různých řešení je k tomuto možno použít například změnu CSS vlastnosti `display` na hodnotu `none.` Poté je již možno použít funkci `elementFromPoint,` pomocí níž je vrácen element, který je na pozici kurzoru myši.

Zvýraznění elementu

Nyní již jen zbývá element na pozici kurzoru vizuálně zvýraznit. Do stránky je při spuštění přidán element s maximální `z-index` hodnotou, který má nastaveno poloprůhledné zelené pozadí. Pokud není používán při vybírání elementu, tak je mu nastavena nulová šířka i výška. Tento element je využit pro zvýraznění elementu na pozici kurzoru tím způsobem, že je mu přiřazena stejná pozice a velikost[20]. Tím, že má vyšší hodnotu z-index CSS vlastnosti, je zobrazen nad ním.



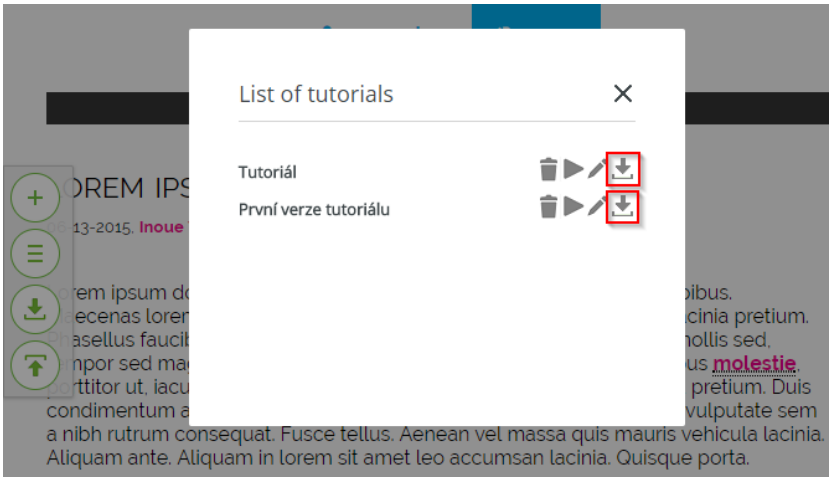
Obrázek 25: Zvýrazněný element, nad nímž je kurzor myši

[17] CSS vlastnost z-index udává prioritu zobrazení při překrývání prvků ve stránce.
[18] Událost `mouseMove` se projeví, pokud se kurzor pohybuje přes plochu elementu.
[19] Funkce `elementFromPoint` vrací element nad daným umístěním ve stránce, který je umístěn nejvýše.
[20] Aktuální pozici a velikost elementu ve stránce je možné zjistit pomocí JavaScript funkce `getBoundingClientRect.`

Tímto postupem je docíleno, že je při výběru ve stránce zvýrazněn vždy ten element, nad nímž je umístěn kurzor myši. Element je ale vybrán až v případě, že je na něj kliknuto. Z toho důvodu je nutné překrývajícímu elementu přiřadit ještě reakci na událost kliknutí. Jelikož víme, který element je aktuálně zvýrazněn, a při kliknutí myší jej můžeme označit za vybraný, není již problém zjistit absolutní cestu vybraného elementu. JavaScript objekt elementu obsahuje jak vlastnosti elementu jako jeho identifikátor a třídy, tak také objekt elementu předka.

5.2.3 Stažení skriptu průvodce z aplikace pro tvorbu

Účelem existence části pro tvorbu je poskytnout uživateli nástroje pro vytvoření průvodce a jeho integraci do webové stránky. Integrace je realizována pomocí skriptu, který je možno stáhnout v uživatelském prostředí aplikace pro tvorbu. Samotný skript průvodce je vytvářen v odděleném projektu a o jeho tvorbě a naplnění daty je hovořeno v části 5.3 Implementace skriptu průvodce.



Obrázek 26: Seznam průvodců s výrazněnými tlačítky pro stažení skriptu průvodce

5.2.4 Uživatelské rozhraní

Skript aplikace pro tvorbu je do stránky vkládán rozšířením prohlížeče. Z toho důvodu je část uživatelského prostředí pro spuštění aplikace vykreslena v nástrojové liště prohlížeče. Po instalaci aplikace v ní přibude ikona aplikace. Při jejím zmáčknutí je ikona změněna a do stránky v aktuálně zobrazené kartě prohlížeče vložen skript pro tvorbu. Ten je v ní automaticky spuštěn. Změna ikony je realizována v event skriptu rozšíření, jak lze vidět na řádku dvě v obrázku 24. Aplikace pro tvorbu je tedy spouštěna pomocí tlačítka v nástrojové liště prohlížeče.

Obrázek 27: Změna ikony aplikace v prohlížeči při jejím spuštění

Uživatelské prostředí aplikace ve stránce je již vykreslováno vloženým skriptem. Toho je docíleno tak, že na konec struktury stránky je pomocí nativní JavaScriptové funkce DOMu `appendChild` vložen element `div`, který má nastavenou CSS vlastnost z-index na maximální kladnou hodnotu 32 bitového čísla. Všechny ostatní části uživatelského rozhraní jsou pak přidávány jako potomci tohoto elementu, díky čemuž jsou vždy zobrazeny nad ostatními elementy stránky a pomocí fixního CSS pozicování jsou umístěny na správném místě ve stránce.

```
▶<div data-jiis="cc" id="main">…</div>
▶<div data-jiis="cc" id="lfootercc">…</div>
 <script>window.gbar&&gbar.up&&gbar.up.tp&&gbar.up.tp();</script>
▶<script>…</script>
▶<div class="gb_6a">…</div>
▼<div id="myTutorial" class="tutorialApp" style="position:absolute;left:0;top:0;z-index:2147483647!important">
   <div id="highlight" style="left: 0px; top: 0px; width: 0px; height: 0px;"></div>
   <div id="pageCover" style="position: fixed; left: 0px; top: 0px; width: 100%; height: 100%; display: block;
   cursor: auto;"></div>
  ▶<div id="mainMenuContainer" style="top: calc(50% - 50px);">…</div>
   </div>
 </body>
</html>
```

Obrázek 28: Část HTML kódu přidaná do stránky aplikací

V rámci implementace programu bylo nutné navrhnout a implementovat jednotlivé části uživatelského rozhraní tak, aby aplikace umožňovala provádět funkce definované v návrhu.

Menu

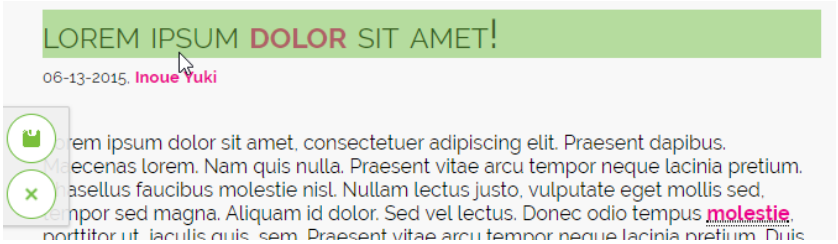
Menu v aplikaci představuje rozcestník, ze kterého je možné přistupovat k jednotlivým funkcím. Po spuštění aplikace pomocí ikony rozšíření je v levé části stránky zobrazeno menu. V menu jsou přístupné vždy jen ty položky, které mají v daný okamžik smysl. Při spuštění aplikace obsahuje položky:

- vytvoření nového průvodce,
- načtení průvodce ze souboru,
- zobrazení seznamu průvodců.

Obrázek 29: Menu vložené aplikací do stránky

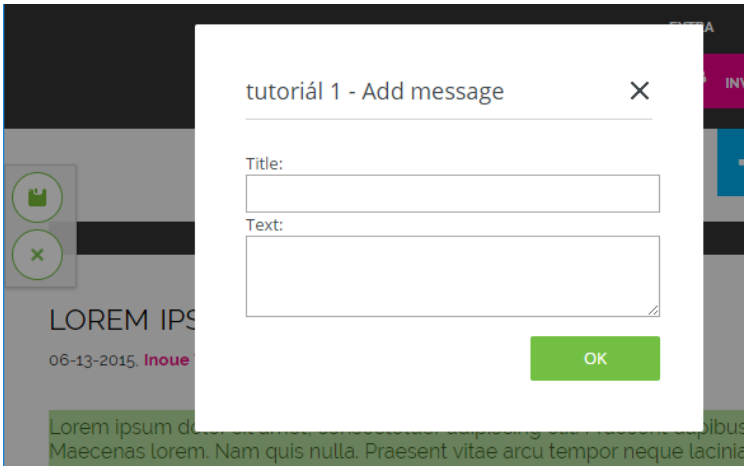
Tvorba průvodce

Start tvorby nového průvodce je možno spustit pomocí tlačítka v menu. Systém zobrazí modální dialog umožňující zadat název průvodce. Následně je aktivovaná funkcionalita samotného výběru elementu stránky. Ten je řešen tak, že se při pohybu kurzoru myši ve stránce zvýrazňuje element nacházející se na nejnižší pozici pod kurzorem. Zvýraznění elementu je řešeno jeho podbarvením do odstínu zelené. V menu jsou v tu chvíli dostupné položky pro uložení průvodce, nebo skončení tvorby bez jeho uložení.



Obrázek 30: Chování aplikace během vybírání elementu

Vybrání elementu probíhá kliknutím na něj myší. V té chvíli je ve stránce zobrazen modální dialog umožňující vložit název a obsah zprávy. Jak je tato funkcionalita řešena technicky, je obsahem následující kapitoly.

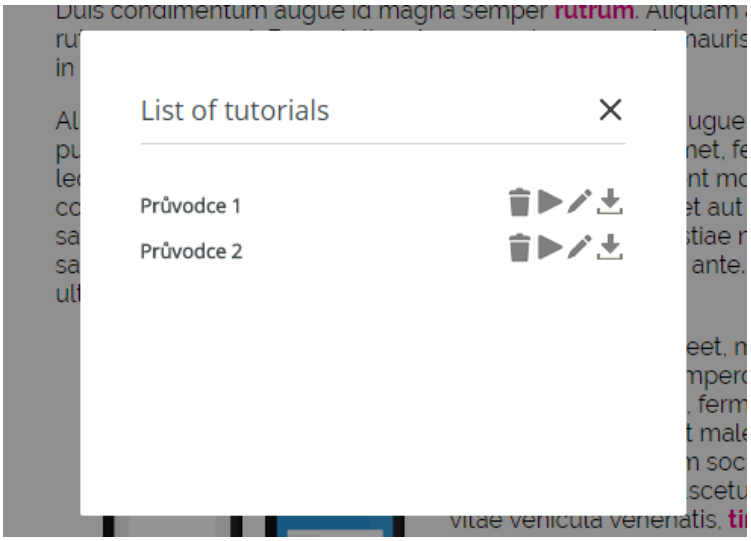


Obrázek 31: Modální okno umožňující zadání obsahu zprávy

Zobrazení seznamu průvodce

Modální okno je spuštěno tlačítkem z menu a je v něm zobrazen seznam průvodců, které byly vytvořeny, nebo načteny se souboru. Z okna je možno spustit následující funkcionalitu:

- smazání průvodce,
- testovací spuštění průvodce ve stránce,
- zobrazení seznamu zpráv průvodce,
- stáhnutí skriptu průvodce.

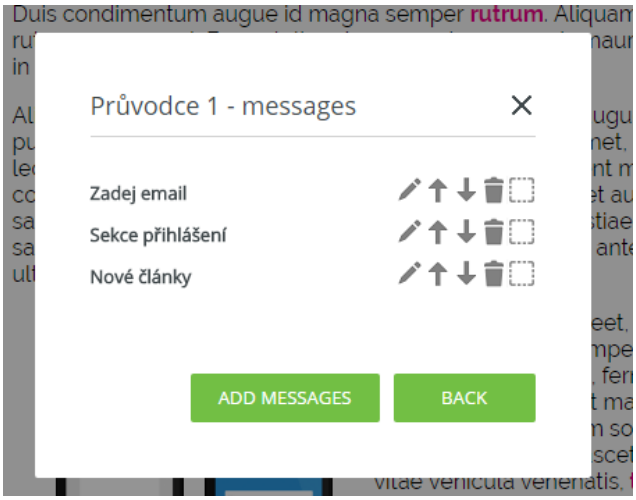


Obrázek 32: Modální okno zobrazující seznam průvodců

Zobrazení seznamu zpráv průvodce

Do tohoto modálního okna je možno se dostat z okna **zobrazení seznamu průvodců**. Obsahuje následující funkcionalitu:

- posun pozice zprávy v průvodci,
- smazání zprávy,
- přidání zprávy do průvodce,
- zobrazení elementu zprávy ve stránce,
- zobrazení okna umožňujícího editovat obsah zprávy.



Obrázek 33: Modální okno zobrazující seznam zpráv průvodce

5.3 Implementace skriptu průvodce

V návrhu bylo rozhodnuto, že aby bylo možné průvodce spustit ve stránce při jejím běžném spuštění v prohlížeči, je nutné jej vyvíjet jako skript. Skript průvodce je stejně jako část pro tvorbu psán v programovacím jazyce TypeScript a do JavaScriptu je transpilován pomocí nástroje BobrilBuild. Výstupem z tohoto projektu je skript v minifikované podobě, který ovšem ještě neobsahuje data průvodce. Ta jsou vytvářena v části pro tvorbu a než je skript stažen uživatelem, jsou do něj data přidána.

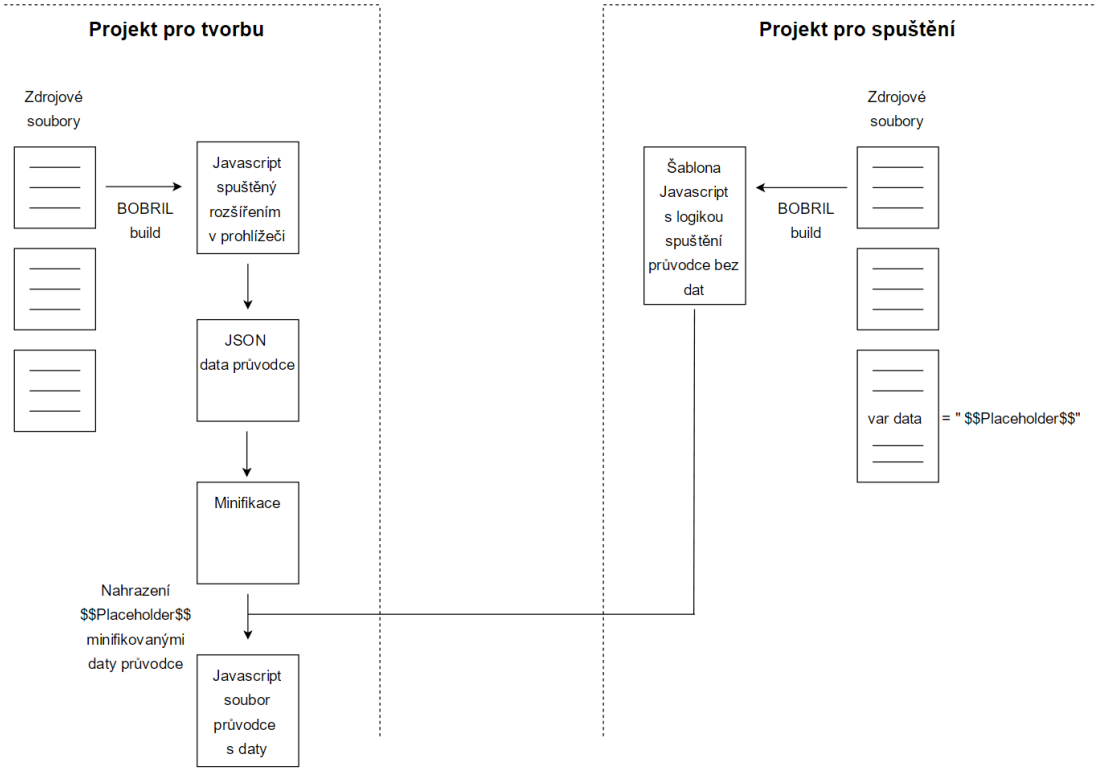
5.3.1 Plnění skriptu průvodce daty z aplikace pro tvorbu

Pro spuštění průvodce ve stránce je nutné do stránky vložit skript, který je možno stáhnout z aplikace průvodce. Ten již obsahuje jak veškerou potřebnou logiku pro zobrazení zpráv, tak data samotného průvodce. Jelikož jsou na sobě oba skripty nezávislé, je nutné si povědět, jakým způsobem je skript pro spuštění průvodce naplněn daty, než je stáhnut z aplikace pro tvorbu.

Skript pro spuštění je tvořen jako samostatný projekt. Výstupem je zminifikovaný,

jednořádkový JavaScript soubor. Tento skript ještě neobsahuje data průvodce. Skript, který je možné stáhnout ze spuštěné aplikace pro tvorbu, již data obsahuje. Toho je docíleno tím způsobem, že skript průvodce očekává data v JSON formátu v textové proměnné. Jelikož názvy proměnné jsou běžně při minifikaci měněny, je zapotřebí místo, kde skript očekává vložení dat, označit jiným způsobem. Tato proměnná má již ve zdrojovém kódu zapsánu hodnotu *$$InsertDataHere$$*. Obsah proměnné není minifikací ovlivněn.

Při požadavku na stažení skriptu průvodce ze spuštěné tvůrčí části je obsah skriptu ze souboru nahrán do textové proměnné. V ní je nalezen text *$$InsertDataHere$$* a ten je poté nahrazen daty průvodce v textovém JSON formátu převedená do jednořádkové reprezentace.



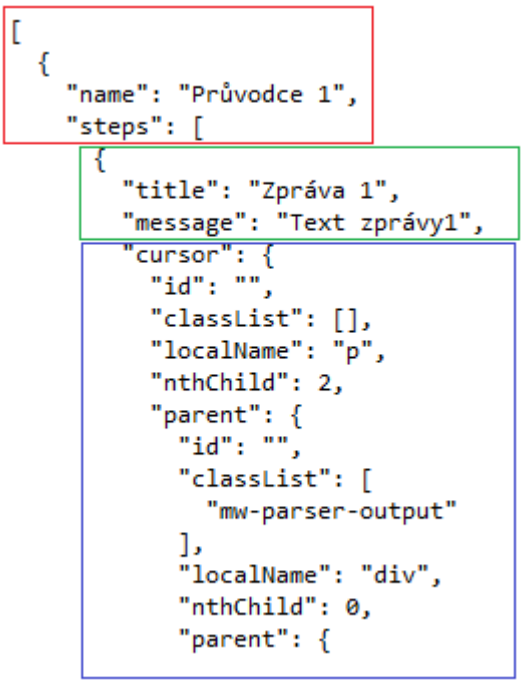
Obrázek 34: Grafické znázornění plnění skriptu průvodce daty

5.3.2 Formát dat průvodce

Aby bylo možné data vytvořená v tvůrčí části aplikace použít v průvodci, bylo zapotřebí udělat rozhodnutí, v jakém formátu budou přenášena. Pro přenos byl zvolen JSON [21]formát. Jedná se o univerzální textový formát založený na syntaxi jazyka JavaScript, který je však na programovacím jazyce zcela nezávislý.

[21] JSON je zkratka pro JavaScript Object Notation.

Jak je vidět na obrázku 35, struktura JSON dokumentu sleduje strukturu entit z ER diagramu na obrázku 16. Červená část reprezentuje průvodce, zelená zprávu a modrá element zprávy včetně rodičovských elementů. Velikost JSONu se bude z velké části odvíjet od toho, jak hluboce byly vybrané elementy ve stránce zanořeny.



Obrázek 35: Část vygenerovaného JSON souboru

Pro generování a parsování hodnot z a do JSON formátu jsou ve skriptech použity nativní funkce prohlížeče, které umožňují dokument generovat z objektů běžící aplikace. Jedná se o funkci `JSON.stringify`. Funkcí, která provádí opačnou úlohu naplnění objektů v běžící aplikaci z textového formátu JSON, je funkce `JSON.parse`.

Stejný formát dat jako pro přenos do spouštěcí části aplikace je využíván také pro funkcionalitu uložení a nahrání rozdělané práce v aplikaci pro tvorbu.

5.3.3 Algoritmus vyhledávání

V rámci návrhu byl představen algoritmus pro vyhledání elementu ve stránce. Jedním z prvních kroků tohoto algoritmu je vyhledání všech elementů, které mají shodný název s elementem hledaným. V pseudokódu hlavní větve programu je to řádek 6. Pro toto je v implementaci využito nativní funkce DOMu `getElementsByTagName`. Tato funkce vrací všechny elementy v dokumentu, které odpovídají zadanému názvu.

Části kódu popsané pseudokódem na obrázcích 20 a 21, které řeší podobnost podle pozice potomka vůči předkovi a podobnost podle tříd, implementují stejné rozhraní `HeuristicsStrategy`.

Jedná se zde tedy o použití návrhového vzoru strategie, neboť kód obou strategií může být měněn a do kódu lze snadno přidávat další strategie [6].

TypeScript kód algoritmu pro vyhledání elementu ve stránce je součástí přílohy B.

5.3.4 Integrace skriptu do stránky

Soubor obsahující skript průvodce, který je stažen z části pro tvorbu, obsahuje tři logické části. První částí je tlačítko. Skript předpokládá, že je tlačítko s identifikátorem startGuideButton umístěno ve stránce a po spuštění je na něj pomocí JavaScript funkce `addEventListener` navázáno spuštění průvodce ve stránce. Toto tlačítko, jenž je na obrázku 36 zobrazeno v prvním řádku, může být umístěno libovolně ve stránce. Jednou z možností je také změna vzhledu pomocí CSS stylování, tak aby zapadlo do grafického stylu stránky. Jediné, co je potřeba neměnit, je jeho identifikátor.

Druhou částí je logika samotného průvodce, která řeší vykreslování zpráv do stránky, přechod mezi zprávami a použití JSON dat. Tato část je na obrázku 36 reprezentována textem za elementem `script`.

```
<button id='startGuideButton' style='color: #fff;background-color: #72bf44;height: 35px;line-heigh
<script>!function(){"use strict";var a=Object.setPrototypeOf||{__proto__:[]}instanceof Array&&func
```

Obrázek 36: Ukázka části souboru staženého z aplikace pro tvorbu

```
window.onload = function (): void {
    document.getElementById("startGuideButton").addEventListener("click", startApplication );
};
```

Obrázek 37: Kód ve skriptu, který naváže spuštění průvodce na tlačítko

Třetí části staženého souboru jsou JSON data průvodce. Ty jsou do skriptu vkládány před jeho stažením v části pro tvorbu způsobem popsaným v kapitole 5.3.1 a jsou obsahem proměnné se kterou je ve skriptu dále pracováno.

```
,f.tutorials[0].playTutorial()}var s='[{"name":"průvodce editorem","steps":[{"title":"Vyplňte telefonní číslo",
```

Obrázek 38: Část skriptu průvodce obsahující JSON data

Poslední částí, o které jsme doposud v souvislostí s integrací skriptu do stránky nehovořili, je způsob, jakým jsou do stránky přidány CSS styly zpráv průvodce. Pomocí nativních JavaScript funkcí DOMu `createElement` a `appendChild` je ve stránce vytvořen nový element `style`, do kterého jsou potřebné styly vloženy. Kód této funkcionality je zobrazen na obrázku 39.

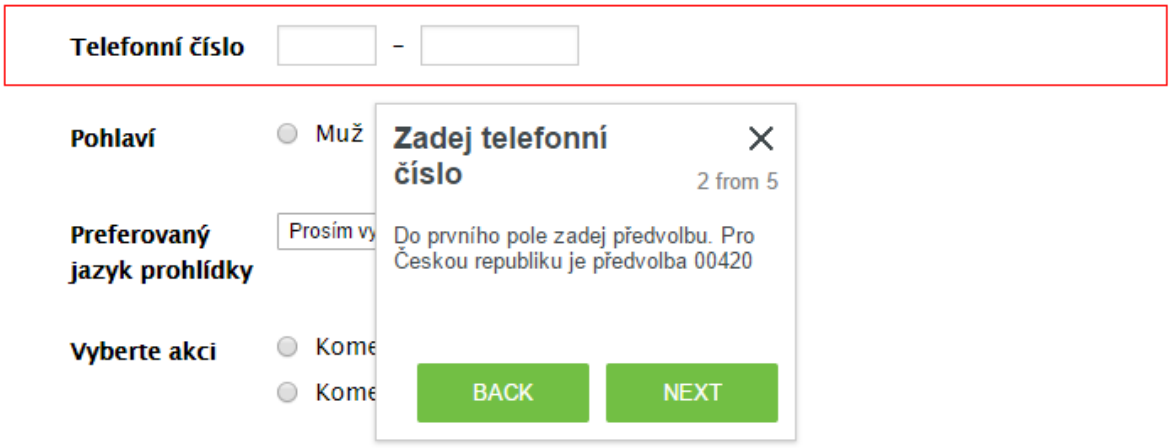
```
static addStyleToPage() {
    let css = `#myTutorial,.tutorialApp .container{color:#40484b;font-family:'Open Sans',sans-serif;font
    let head = document.head;
    let style = document.createElement('style');
    style.textContent = css;
    style.type = 'text/css';
    head.appendChild(style);
}
```

Obrázek 39: Vložení CSS stylů do stránky

Na umístění skriptu ve stránce nezáleží. Může být umístěn jak v hlavičce, tak v elementu `body`, jen je třeba zajistit, aby byl při spuštění skriptu ve stránce přítomný element s identifikátorem startGuideButton.

5.3.5 Uživatelské rozhraní

Uživatelské rozhraní spuštěného průvodce je tvořeno zprávami, mezi kterými je možno procházet směrem dopředu i dozadu. Zpráva obsahuje titulek a obsah. Dále je v ní uvedeno, na které zprávě z průvodce se aktuálně nacházíme. Průvodce lze kdykoli ukončit pomocí tlačítka v pravém horním rohu zprávy.



Obrázek 40: Zobrazená zpráva průvodce

Technicky je zobrazování zpráv ve stránce řešeno velice podobně jako v případě uživatelského rozhraní části pro tvorbu. Zprávy jsou do struktury stránky přidávány pomocí nativní JavaScriptové funkce DOMu `appendChild`.

5.4 Shrnutí

V kapitole je popsáno, z jakého důvodu byl pro implementaci zvolen jazyk TypeScript. Dále je popsáno, co je to rozšíření prohlížeče a jakým způsobem bylo využito v realizované aplikaci. Poté je zvlášť představeno řešení pro tvorbu a řešení pro spuštění a dále je vysvětleno, jakým způsobem jsou

přenášena data mezi jednotlivými částmi.

Výstupem fáze implementace je funkční řešení umožňující vytvořit pomocí přehledného uživatelského rozhraní průvodce pro libovolnou webovou stránku. Část aplikace, která umožňuje tvorbu a editaci, je distribuována jako rozšíření do prohlížeče Chrome, část pro spuštění je realizována skriptem.

6 Srovnání řešení s existujícími programy

Oproti vyvíjené aplikaci mají všechny nástroje představené v kapitole 3 jednu rozdílnost, která by se za určitých podmínek dala považovat za výhodu, při jiných za nevýhodu. Skript určený k integraci průvodce do stránky u všech výše zmíněných aplikací neobsahuje logiku pro zobrazení zpráv ani data průvodce. Vše je skriptem stahováno ze serveru. Výhodou tohoto přístupu je, že po úpravě tutoriálu není nutné do stránky nahrávat pozměněný skript. Ten je stále stejný, pozměněná data jsou pouze na serveru. Nevýhodou v tomto případě ovšem je, že spuštění tutoriálu je závislé na dostupnosti serveru aplikace průvodce.

Naproti tomu jsou u aplikace vyvinuté v rámci diplomové práce jak uživatelské rozhraní, tak jednotlivé kroky průvodce součástí skriptu. Toto řešení umožňuje aplikaci použít také pro případy, kdy stránka běží pouze lokálně na počítači, který je bez přístupu k internetu. Dnes je také možno vyvíjet desktopové aplikace pomocí webových technologií http, JavaScript a CSS. Vyvíjené řešení je vhodné také pro takovéto aplikace. Vlastností tohoto řešení je potřeba opětovného nahrání skriptu, pokud chceme aplikovat změnu v průvodci, a také větší velikost vkládaného skriptu. Nejedná se ovšem o klíčovou architektonickou vlastnost a změna způsobu vložení skriptu do stránky je snadno modifikovatelná.

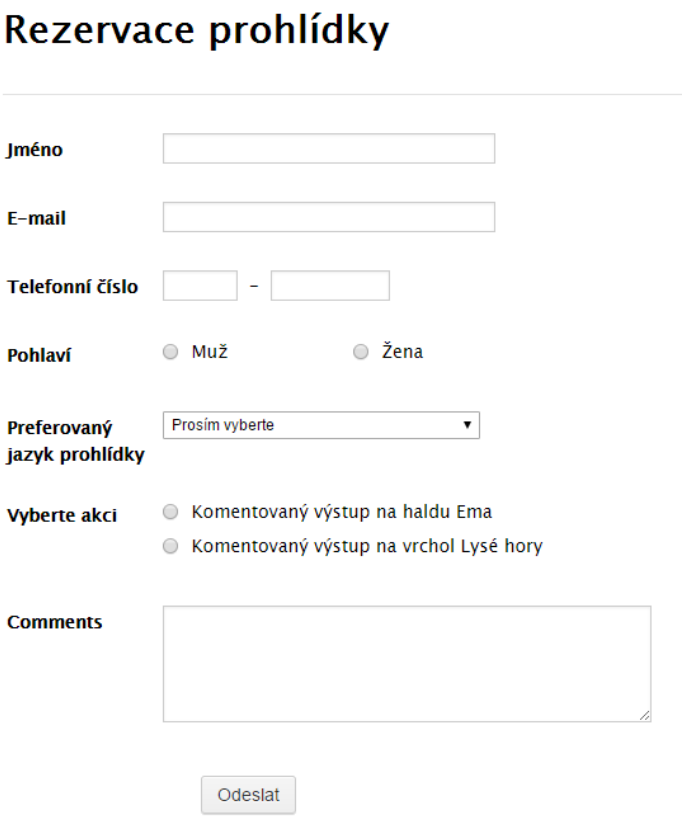
7 Případová studie

Případová studie je jednou z metod kvalitativního výzkumu. Stručně bývá charakterizována jako detailní studium jednoho, či malého počtu případů za účelem aplikace získaných poznatků při porozumění případům obdobným [5]. V případové studii ukážeme, jakým způsobem je možné použít naimplementovanou aplikaci pro řešení reálného problému.

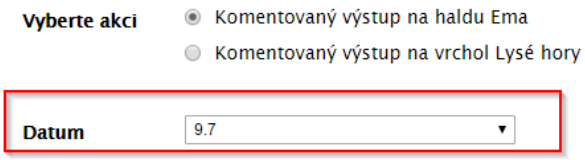
7.1 Vstup

Případová studie ukáže možnost využití implementovaného řešení pro rozšíření webových stránek. Konkrétně se bude jednat o stránky, které podporují podnikatelskou aktivitu pořádání komentovaných prohlídek v Moravskoslezském kraji. Součástí těchto stránek je také registrační formulář, umožňující se lidem registrovat na konkrétní prohlídku. Jelikož nám na prohlídky chodí také velké množství starších lidí, u kterých je předpoklad, že nemusí být vždy počítačově příliš gramotní, budeme chtít formulář vylepšit o možnost jej vyplnit za pomoci průvodce. Pro tvorbu průvodce využijeme aplikaci vyvinutou v rámci řešení diplomové práce.

Formulář je zobrazen na obrázku 41. Tento registrační formulář chceme rozšířit o průvodce, který uživatele provede vyplňováním jednotlivých položek. Všechny položky však nejsou zobrazovány ihned při spuštění stránky. Položka datum je zobrazena až ve chvíli, kdy je vybrána akce.



Obrázek 41: Formulář rezervace prohlídky



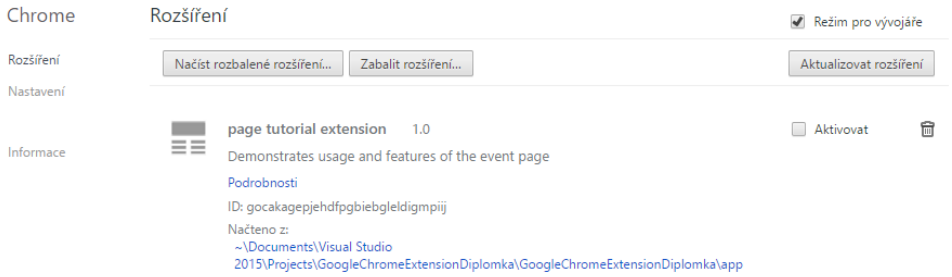
Obrázek 42: Položka přidaná do formuláře až poté, kdy je vybrána akce

7.2 Tvorba průvodce a integrace do stránky

Instalace nástroje pro tvorbu

Prvním krokem, který bylo třeba ověřit a udělat, než jsme s aplikací mohli začít pracovat, bylo ji dle návodu uvedeného v příloze nainstalovat do prohlížeče Google Chrome. V prohlížeči v nástrojové liště přibyla ikona pro spuštění aplikace ve stránce a v panelu rozšíření prohlížeče přibylo nainstalované rozšíření.

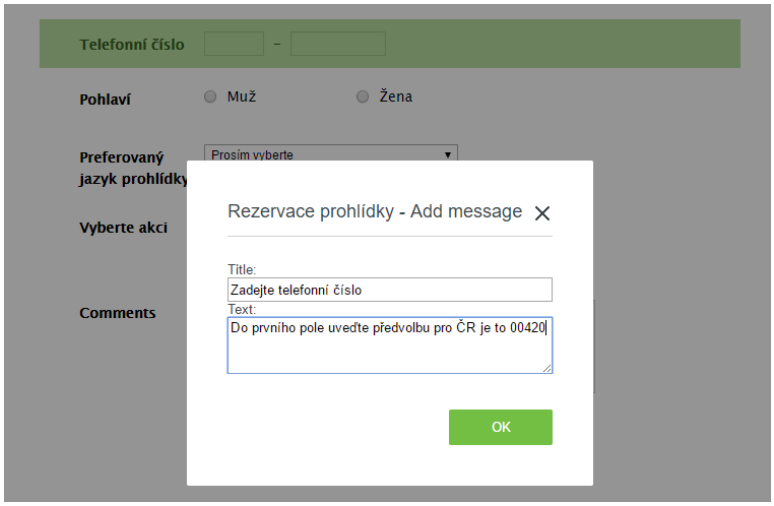
Obrázek 43: Ikona aplikace v prohlížeči Google Chrome



Obrázek 44: Aplikace zobrazena v panelu rozšíření

Tvorba průvodce

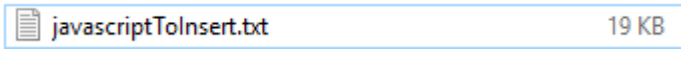
Pro každé pole v registračním formuláři prohlídky byla vytvořena jedna zpráva. Aby bylo možné ve formuláři přiřadit zprávu k položce datum, bylo před tímto krokem zapotřebí vybrat v tomtéž formuláři položku *vyberte akci*. Mezi tvorbou jednotlivých zpráv lze se stránkou běžně pracovat, takže v ní lze provádět stejné kroky, jako se požadují po uživateli.



Obrázek 45: Tvorba zprávy

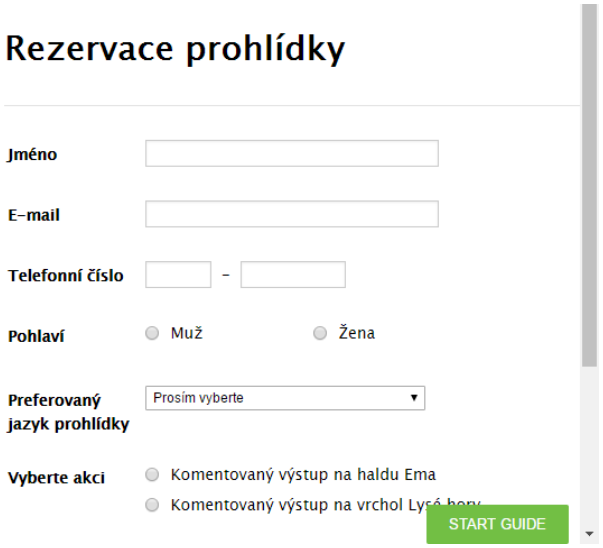
Stažení skriptu průvodce

V rozhraní aplikace tvorby průvodce byl stažen skript pro integraci do stránky. Velikost tohoto skriptu je 19 kilobajtů. Soubor obsahuje jak samotný skript, tak tlačítko v HTML formátu.



Integrace skriptu do stránky

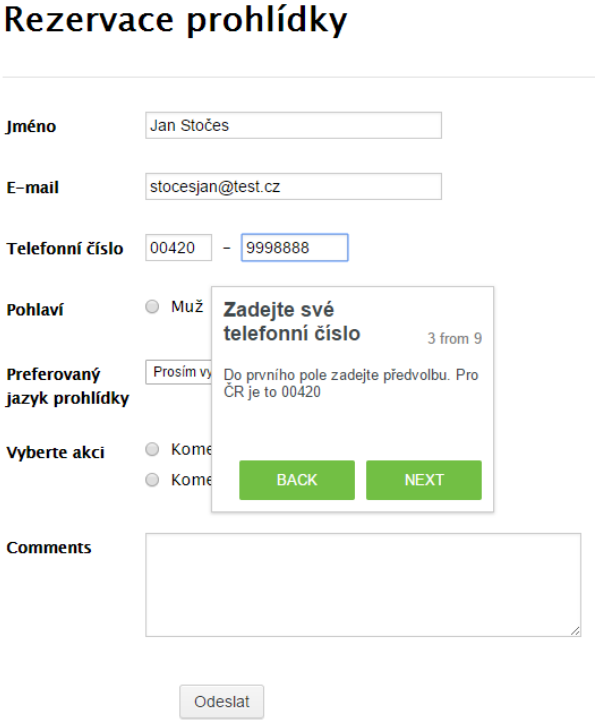
Obsah skriptu byl vložen do stránky a takto pozměněná stránka byla nahrána na server místo původní. Po otevření stránky v prohlížeči v ní přibylo tlačítko zobrazené v pravém dolním rohu.



Obrázek 46: Stránka spuštěná poté, co do ní byl nahrán skript

Spuštění

Po kliknutí na přidané tlačítko je ve stránce spuštěn průvodce. Průvodce je procházen krok po kroku. Mezi jednotlivými kroky jsou ve formuláři vyplňovány položky podle instrukcí ze zpráv.



Obrázek 47: Průvodce spuštěný ve stránce

7.3 Zhodnocení

Na tomto případu bylo ověřeno, že je aplikaci možno použít pro tvorbu průvodce na reálné stránce, tak aby podporovala její funkcionalitu. Jak tvorba, tak nasazení proběhly bez problémů a zabraly méně než 10 minut času. Vygenerovaný skript měl velikost 19 kilobajtů, což není mnoho a nepředstavuje to velkou zátěž při stahování stránky ze serveru do prohlížeče. Při procházení formuláře krok za krokem a plněním instrukcí, které jsou v něm zadány, byly všechny zprávy postupně zobrazeny a správně umístěny u položek, které mají popisovat.

8 Závěr

Cílem této práce bylo vyvinout samostatný systém umožňující realizaci interaktivních průvodců v rámci obecných webových prezentací s důrazem na nezávislost na hostitelské aplikaci. Cíle práce byly naplněny a výsledkem je návrh a realizace řešení, které umožňuje jak tvorbu průvodce, tak jeho integraci do libovolné webové stránky.

První část práce je věnována popisu toho, co pojem interaktivní průvodce webovou aplikací znamená a jak je v praxi používán. Následuje přehled tří již existujících programů, které tvoří typické zástupce tohoto segmentu trhu, u kterých je popsána jejich funkcionalita a technické řešení. V další části je rozebírán návrh architektury a koncepční řešení aplikace průvodce s důrazem na obecnou použitelnost algoritmu vyhledání elementu ve stránce i při dílčích úpravách obsahu stránky, což je důležitou vlastností pro praktické využití aplikace. V části implementace jsou popsány implementační detaily práce, tedy v jakém jazyce je aplikace napsána, jakým způsobem jsou části aplikace technicky řešeny, jak je distribuována a jakým způsobem je pracováno s vytvořeným průvodcem webovou aplikací. Závěrečná část práce, kterou je kapitola případ použití, popisuje celý životní cyklus vytvořené aplikace od vytvoření průvodce až po jeho integraci do zvolené internetové stránky.

Hlavním přínosem práce je vytvoření uceleného ekosystému aplikace pro tvorbu průvodců webovými stránkami, který je ve srovnání s jediným představeným open source řešením WalkHub podstatně snadnější na používání. Dalším důležitým přínosem práce je návrh a implementace algoritmu pro lokalizaci elementu na stránce, který je funkční i při dílčích změnách stránky a jeho implementace za použití návrhového vzoru Strategie usnadňuje budoucí úpravy a rozšíření.

Možnosti rozšíření aplikace do budoucna vidím hlavně ve vývoji dalších strategií identifikace prvku na stránce (např. obsah sousedních prvků, pozice na vykreslené stránce atp.) a dalších možností distribuce průvodce, např. místo vložení do cílové stránky pouze vložení odkazu.

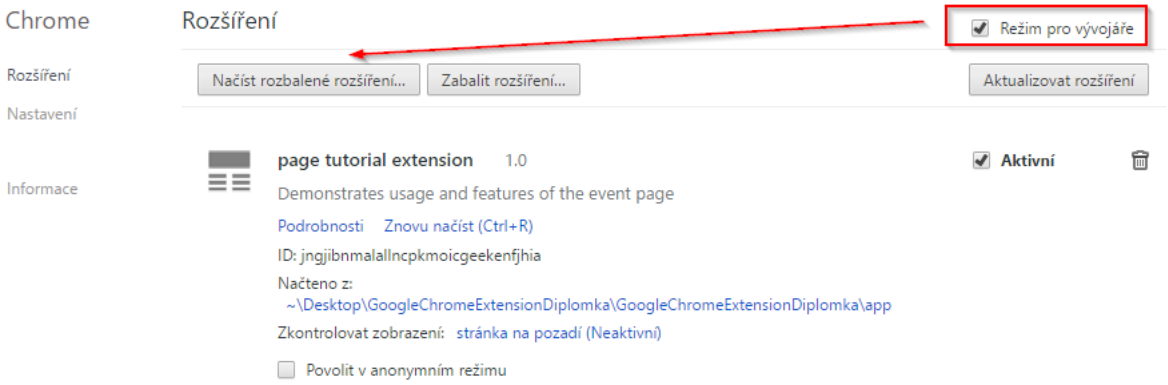
Seznam literatury a informačních zdrojů

1. VONDRÁK, I. Úvod do softwarového inženýrství. Ostrava: VŠB, 2002.
2. ROZENTALS, N. Mastering TypeScript. London: Packt Publishing, 2015.
3. *BobrilBuild. Introduction* [online]. [cit. 2017-06-26]. Dostupné z: <www.codeproject.com/Articles/1167901/Bobril-V-Bobril-build>.
4. *Content Scripts. Chrome extensions* [online]. [cit. 2017-07-02]. Dostupné z: <https://developer.chrome.com/extensions/content_scripts>.
5. HENDL, J. Úvod do kvalitativního výzkumu. Praha: Karolinum, 1997.
6. FREEMAN, E. – FREEMAN, E. – SIERRA, K. – BATES, B. Head First design patterns. Sebastopol: O'Reilly, 2004.
7. HALES, W. HTML5 and JavaScript web apps. Sebastopol: O'Reilly, 2012.
8. Jason Beaird: The Principles of Beautiful Web Design, SitePoint, 2014.
9. *TypescriptLang: Documentation* [online]. [cit. 2017-06-15]. Dostupné z: <https://www.typescriptlang.org>.
10. *ChromeExtensions: Development* [online]. [cit. 2017-06-24]. Dostupné z: <https://developer.chrome.com/extensions>.
11. Erixc Elliot: Programming Javascript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries, O'Reilly Media, 2014.
12. Rick Delorme: Programming in HTML5 with JavaScipt and CSS3, Redmont:MicrosoftPress, 2014.

Přílohy

A Instalace aplikace pro tvorbu

Aby mohla být aplikace dostupná z oficiálního obchodu prohlížeče Chrome, je nutné mít vytvořen placený účet u společnosti Google. Z tohoto důvodu je aplikaci nutné nainstalovat pomocí takzvaného režimu vývojáře, který je nutné zapnout v kartě rozšíření prohlížeče Chrome. Poté je na tuto kartu přidáno tlačítko načíst rozbalené rozšíření. Poté již jen stačí vyhledat adresář obsahující soubory rozšíření a aplikace je automaticky nainstalována.



Poté, co je aplikace nainstalována do prohlížeče, je do nástrojové lišty přidána ikona a po jejím zmáčknutí je aplikace ve stránce spuštěna a je možné začít vytvářet průvodce.

B Kód algoritmu pro vyhledávání elementu ve stránce

```
class ElementFinding {
    static findElement(cursor: IElementCursor): HTMLElement {
        // quick search by ID
        let byId = document.getElementById(cursor.id);
        if (byId != null) {
            return byId;
        }
        // find candidate elements by tag name (div, ul, li...)
        let elementsByTag = document.getElementsByTagName(cursor.localName);
        // create cursors from elements
        let candidateCursors = ElementFinding.createCursors(elementsByTag);
        // keep only candidate cursors with longest prefix
        candidateCursors = ElementFinding.pruneShorterPrefixes(cursor, candidateCursors);

        if (candidateCursors.length == 0) {
            // not found;
            return null;
        } else if (candidateCursors.length == 1) {
            // hurray
            return candidateCursors[0].element;
        } else {
            // multiple candidates found, apply heuristics
            return ElementFinding.applyHeuristics(cursor, candidateCursors)
        }
    }
```

```
private static applyHeuristics(cursor: IElementCursor, candidateCursors: IElementCursor[]): HTMLElement {
    // look for full match in order
    let childPosStrategy = new ChildPositionStrategy();
    for (let i = 0; i < candidateCursors.length; i++) {
        let score = childPosStrategy.getScore(cursor, candidateCursors[i]);
        // full order match
        if (score == 1) {
            return candidateCursors[i].element;
        }
    }
    // no full position match found -> apply position + classes strategy
    let classesStrategy = new ClassesStrategy();
    let highestScore = 0;
    let highestScoredElement = null;
    // find element with highest heuristic score
    for (let i = 0; i < candidateCursors.length; i++) {
        let posScore = childPosStrategy.getScore(cursor, candidateCursors[i]);
        let classesScore = classesStrategy.getScore(cursor, candidateCursors[i]);
        // sum scores (can also weight each strategy differently)
        if ( posScore + classesScore > highestScore) {
            highestScore = posScore + classesScore;
            highestScoredElement = candidateCursors[i].element;
        }
    }
    // return highest scored element (or NULL if all heuristics were 0)
    return highestScoredElement;
}

private static pruneShorterPrefixes(cursor: IElementCursor, candidateCursors: IElementCursor[]) {
    let longestPrefix = 0;
    let prefixed = [];
    candidateCursors.forEach((c, index) => {
        let prefixLength = this.calculateCommonPrefixLength(cursor, c);
        // keep track of longest common prefix
        if (prefixLength > longestPrefix) {
            longestPrefix = prefixLength;
        }
        prefixed[index] = prefixLength;
    });
    // return variable
    let prunedCandidates = [];
    // put to return array only cursors with longest common prefix
    for (var i = 0; i < prefixed.length; i++) {
        if (prefixed[i] == longestPrefix) {
            prunedCandidates.push(candidateCursors[i]);
        }
    }
    // return pruned candidates
    return prunedCandidates;
}

// calculate common prefix length for two cursors
// TODO not so sure this works
private static calculateCommonPrefixLength(c1: IElementCursor, c2: IElementCursor) {
    let same = (c1 && c2) && c1.localName == c2.localName;
    return same ? (1 + this.calculateCommonPrefixLength(c1.parent, c2.parent)) : 0;
}

// create cursors from list of elements
private static createCursors(elements: NodeListOf<Element>): IElementCursor[] {
    let cursors = [];
    for (var i = 0; i < elements.length; i++) {
        cursors.push(getElementCursor(<HTMLElement>(elements.item(i))));
    }
    return cursors;
}
```

```
/**
 * Heuristic strategy is applied to cursor (element we are looking for) and candidate (element we want to evaluate)
 * Strategy returns number in interval <0,1> where 0 is no match, 1 is full match
 *
 */
interface HeuristicStrategy {
    getScore(cursor: IElementCursor, candidate: IElementCursor): number;
}


class ChildPositionStrategy implements HeuristicStrategy {

    getScore(cursor: IElementCursor, candidate: IElementCursor): number {
        // sum of order differences
        let sum = 0;
        while (cursor && candidate) {
            sum += Math.abs(cursor.position - candidate.position);
            cursor = cursor.parent;
            candidate = candidate.parent;
        }
        // sum == 0 -> perfect order match
        if (sum == 0) {
            return 1;
        } else {
            // sum > 0 -> higher the diff, closer to zero
            return 1 / (1 + sum);
        }

    }

}

class ClassesStrategy implements HeuristicStrategy {

    getScore(cursor: IElementCursor, candidate: IElementCursor): number {
        // sum of class differences
        let sum = 0;
        while (cursor && candidate) {
            sum += this.calculateArrayDiff(cursor.classList, candidate.classList);
            cursor = cursor.parent;
            candidate = candidate.parent;
        }
        // sum == 0 -> perfect classes match
        if (sum == 0) {
            return 1;
        } else {
            // sum > 0 -> higher the diff, closer to zero
            return 1 / (1 + sum);
        }
    }

    private calculateArrayDiff(list1: string[], list2: string[]) {
        let total = 0;
        let common = 0;
        let set = [];
        for (let i = 0; i < list1.length; i++) {
            set[list1[i]] = 1;
            total++;
        }
        for (let i = 0; i < list2.length; i++) {
            if (set[list2[i]]) {                                    //nebylo [i]
                common++;
            } else {
                total++;
            }
        }
        return total - common;
    }
}
```

C Obsah přiloženého CD

Na přiloženém CD se nachází zdrojové kódy vyvinutého systému, tato diplomová práce v elektronické podobě, nástroj pro tvorbu, zdrojové soubory skriptu tvůrce a zdrojové soubory skriptu průvodce, HTML stránka použitá v případové studii. V následující tabulce je uvedena struktura přiloženého CD.

Adresář	Popis
/ GuideCreatorChromeExtension	Chrome rozšíření připravené k nainstalování podle návodu v příloze A
/Src/GuideCreatorProject	Zdrojové soubory skriptu pro tvorbu
/Src//GuideLauncherProject	Zdrojové soubory skriptu pro spuštění
/ FormWithGuide.html	Stránka s formulářem obsahující průvodce
/ FormWithoutGuide.html	Stránka s formulářem neobsahující průvodce
/ tutorials.json	JSON data připravená k načtení do aplikace pro tvorbu